1244.42977X00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): T. A. GRAY, et al

Serial No.:

Filed: August 1, 2003

Title: ARCHITECTURE AND IMPLEMENTATION FOR CONTROL OF CONTEXT AWARE CALL PROCESSING WITH LOCAL FEATURE DEFINITION

Group:

LETTER CLAIMING RIGHT OF PRIORITY

Mail Stop Patent Application
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

August 1, 2003

Sir:

Under the provisions of 35 USC 119 and 37 CFR 1.55, the applicant(s) hereby claim(s) the right of priority based on Great Britain Patent Application No.(s) 0218713.6 filed August 12, 2002.

A certified copy of said United Kingdom Application is attached.

Respectfully submitted,

ANTONELLI, TERRY, STOUT & KRAUS, LLP

Donald E. Stout
Registration No. 26,422

DES/nac
Attachment
(703) 312-6600

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.



Signed P. Mahoney

Dated 2 July 2003

Patents Form 1/77

Patents Act 1977
(Rule 16)

The Patent Office

13AUG02 E740396-16 C65849
P01/7700 0.00-0218713.6

1/77

12 AUG 2002
LONDON

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office

Cardiff Road
Newport
South Wales
NP10 8QQ

1. Your reference 1057-0003

2. Patent application number *(The Patent Office will fill in this part)* 0218713.6 12 AUG 2002

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

Mitel Knowledge Corporation
350 Legget Drive
Ottawa, Ontario
K2K 2W7 Canada

Patents ADP number *(if you know it)* 08090557003

If the applicant is a corporate body, give the country/state of its incorporation Canada

4. Title of the invention

Architecture and Implementation for Control of Context Aware Call Processing with Local Feature Definition

5. Name of your agent *(if you have one)*

"Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

IPULSE
26 Mallinson Road,
London SW11 1BP
UK

Patents ADP number *(if you know it)* 07929530001 07977754001

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number	Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application	Number of earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*

a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.
See note (d))

YES

tents Form 1/77

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	NIL
Description	36
Claim *(s)*	2
Abstract	1
Drawing *(s)*	20 +28

10. If you are also filing any of the following, state how many against each item.

Priority documents	0
Translations of priority documents	0
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	1
Request for preliminary examination and search *(Patents Form 9/77)*	1
Request for substantive examination *(Patents Form 10/77)*	0
Any other documents *(please specify)*	

11. I/We request the grant of a patent on the basis of this application.

Signature: IPDLS

Date: 12 August 2002

12. Name and daytime telephone number of person to contact in the United Kingdom: David Rickard 020 7223 4979

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) If you need help to fill in this form or you have any questions, please contact the Patent Office on 08459 500505.

b) Write your answers in capital letters using black ink or you may type them.

c) If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.

d) If you have answered 'Yes' Patents Form 7/77 will need to be filed.

e) Once you have filled in the form you must remember to sign and date it.

f) For details of the fee and ways to pay please contact the Patent Office.

ents Form 7/77

ents Act 1977
(Rule 15)

7/77

Statement of inventorship and of right to grant of a patent

12 AUG 2002

The Patent Office

Cardiff Road
Newport
South Wales
NP10 8QQ

1. Your reference 1057-0003

2. Patent application number *(if you know it)* 0218713.6

12 AUG 2002

3. Full name of the or of each applicant

Mitel Knowledge Corporation

4. Title of the invention

Architecture and Implementation for Control of Context Aware Call Processing with Local Feature Definition

5. State how the applicant *(s)* derived the right from the inventor *(s)* to be granted a patent

The inventors were at the relevant time employees of the applicant and the applicant is entitled by virtue thereof.

6. How many, if any, additional Patents Forms 7/77 are attached to this form? *(see note (c))* NIL

7. I/We believe that the person *(s)* named over the page *(and on any extra copies of this form)* is/are the inventor *(s)* of the invention which the above patent application relates to.

Signature

Date 12 August 2002

8. Name and daytime telephone number of person to contact in the United Kingdom David Rickard 020 7223 4979

Notes

a) *If you need help to fill in this form or you have any questions, please contact the Patent Office on 08459 500505.*

b) *Write your answers in capital letters using black ink or you may type them.*

c) *If there are more than three inventors, please write the names and addresses of the other inventors on the back of another Patents Form 7/77 and attach it to this form.*

d) *When an application does not declare any priority, or declares priority from an earlier UK application, you must provide enough copies of this form so that the Patent Office can send one to each inventor who is not an applicant.*

e) *Once you have filled in the form you must remember to sign and date it.*

Enter the full names, addresses and postcodes of the inventors in the boxes and underline the surnames

LISCANO, Ramiro
27 Nanaimo Drive
Ottawa, Ontario
K2H 6X9 Canada

0813814100

Patents ADP number *(if you know it):*

GRAY Thomas A.
111 Pineridge Road
RR3 Carp, Ontario
Canada K0A 1L0

0625085002

Patents ADP number *(if you know it):*

Patents ADP number *(if you know it):*

Reminder

Have you signed the form?

DUPLICATE

Architecture and Implementation for Control of Context Aware Call Processing with Local Feature Definition

Field of the Invention

This invention relates generally to Internet telephony, and more particularly to a call processing architecture and feature model adapted for implementation via the Internet.

Background of the Invention

Inexpensive connectivity provided by the Internet, along with wireless devices and the emergence of ubiquitous sensors is creating the conditions for a generation of new services based on an 'always-connected' feature model. In this model users are always reachable by means of portable wireless appliances without limitation due to time or location. Conversely, users are also always able to reach useful collaborators and information. These capabilities call into question a number of assumptions that underlie traditional prior art call processing models. Many features of prior art call processing models make implicit assumptions about system and user behavior that are used to handle calls to a busy or not-present user. In the 'always-connected' feature model, the 'not-present' assumption is invalidated.

Traditional models of call processing fall into two distinct categories. The older model utilizes centralized call processing in which a call is created as a single entity in a single place. In such prior art systems, 'busy' meant that a telephone set was off-hook and therefore the telephone switch control could not communicate with it.

In the newer Intelligent Network (IN) SS7 (signaling system 7) model, the call results from the cooperation of two distinct entities (call-halves) which cooperate by means of message passing and negotiation over a digital backbone using ISUP protocol. Feature logic is centralized in offices referred to as SCPs (service control points). Features operating in the local offices (SSPs or service signaling points) may have trigger points in their code for referring operation to an SCP to obtain proper behavior.

The distributed call-halve model allows for call processing to be physically distributed. With call state sharing, the assumptions regarding 'busy' status have been modified giving rise to features such as 'call waiting' and 'camp on'. A call process separate from the equipment used to make call connection has allowed for communication with the user on an existing call, and user notification of a call queue awaiting his/her attention.

Both of these call-processing models share the characteristic of a centralized definition of features. Indeed, in the instantiation of the Intelligent Network (IN), this was considered to be a primary benefit of the distributed model (i.e. new features can be defined and provisioned centrally in the IN). This has resulted in the benefit of creating a faster time to market for new features and greater stability in the network as features need to be designed only once to run on switching systems purchased from any manufacturer. Network stability has also been enhanced since features have been designed by teams of experienced designers with a view to avoiding 'bugs' resulting form interactions of multiple feature designs. Most useful features are specific to and have knowledge of the current user context. IN services therefore have great difficulty with customized and personalized services, which are the services of most utility. As a result the IN and its successor the Advanced Intelligent Network (AIN) have not been able to achieve what had been hoped.

The Internet has created an emphasis on a new set of non-functional requirements that have been difficult to achieve using the above-described existing models of call processing. Centralized feature definitions make it difficult to create customized services tailored to particular enterprises in the public network and to create services personalized to the needs of specific users in enterprise networks, which are hallmarks of Internet applications.

More recently, the SIP and CPL protocols have been developed by IETF to provide user defined call handling (see Sinnreich, H; Johnston, A.B., *Internet Communication Using SIP*, Wiley, 2001 pp 104-108, and The Internet Draft by Schulzrinne, H.; Rosenberg, J.; SIP Caller Preferences and Callee Capabilities, draft-ietf-sip-callerprefs-05.txt available at: http://www.softarmor.com/sipwg/drafts/draft-ietf-sip-callerprefs-05.txt). In these models call disposition is not controlled by a centralized set of standard

features designed in a lab environment by expert developers. Rather, the user specifies a set of personal features that can be tailored to closely match the needs of the user's activities within an enterprise environment. These personal features can be cognizant of the identity of the user's collaborators, the nature of the work being done, the user's calendar and so on.

Within SIP, a user may register several Contact addresses with the proxy server that is serving him/her. The user may specify in the SIP INVITE message the types of connections he/she will accept. This is done by specifying the characteristics of the end point (or User Agent) in the ACCEPT-CONTACT and REJECT-CONTACT headers that the user prefers to either accept or reject, respectively. These headers can specify specific end-points in terms of their URLs as being preferred for acceptance or rejection.

The ability to identify specific end points that are preferred for rejection or acceptance indicates a new level of features that are being enabled by SIP. The caller preferences in SIP may specify specific characteristics of the physical device at the end point, which provides an easily understood application of this protocol to conventional telephony.

However, there is no adequate explanation in the SIP drafts of how conflict between called party and caller preferences may be identified and resolved. The CPL specification [Lennox, J.; Schulzrinne, H.; Call Processing Language Framework and Requirements, IETF RFC2824, http://www.ietf.org/internet-drafts/draft-ietf-iptel-cpl-06.txt] mentions that certain caller preferences may be ignored in making a decision. However no guidance is given as to whether such a capability is acceptable or even useful. In short the currently specified SIP call handling procedures are inadequate for call handling in realistic call processing situations.

Thus, SIP and CPL do not provide an effective architecture for handling user preferences in the execution of features and especially in the handling of conflicts between preferences. Consequently, doubts have been expressed concerning the scalability and evolvability of such protocols to realistically sized systems using Internet technology.

Accordingly, the inventor has recognized the desirability of extending the IETF model, understanding the features that it enables and developing practical technological solutions to issues in implementation of these features.

Summary of the Invention

In the present invention, an Active Call Delivery (ACD) system is provided according to a distributed call processing model which uses local rather than centralized feature definition. Communication end points within the system have the ability to define their own feature behavior. A standard language is used to convey indications between end points for negotiation of call behavior. There are two important differences between the call processing model of the present invention and prior art call processing models. Firstly, there is no distributed call state. This follows from the capability of each communication end point to define its own call handling preferences, such that the call states of each end point are in general unknowable to each other end point (i.e. in Internet telephony there is no central repository of features). Secondly, the indications exchanged between each side have no commonly shared semantics. They are merely syntactic entities that a receiving end point can interpret as indications of the goals of the sender end point for the purpose of creating an interaction that meets the needs of its user.

Several new applications in call processing are possible in the system according to the present invention that are characterized by the interpretation of external events to facilitate personalized and customized services. There are three main types of such new application implemented services.

Personalized call control services are provided which are aware of the user's relationships in his/her business environment. These services interpret the importance and urgency of a call in the context of the relationship between the called and calling parties. For example, this type of service processes information to determine if a call is relevant to the user's current activities or if it is from an important customer, etc. to determine how it should be handled.

A second type of service shares awareness of a user's availability. Instead of controlling incoming calls and restricting access as traditional telephony does, this type of

service attempts to engender useful communication by advertising the availability of potential collaborators. In order to do so, it attempts to understand the user's current area of interest either by taking direct instruction from the user or by interpreting his/her activities. It then determines the interest of potential collaborators in this area of interest and presents that information to the user. This use of availability can be extended to directly selecting expert collaborators and other resources.

The third type of new service utilizes events in the enterprise environment to make the user aware of activities of his/her collaborators. This includes examples such as notification and alert services. For example, the registering of a visitor at the reception area of an enterprise may be taken as an indication that an alert should be sent to a user's pager if he/she is not at his/her desk. This action happens automatically without human intervention. Additional examples include airplane schedule changes and stock price alerts, etc. The above examples exist in a virtual space that models the user's activities. However other examples can be given in which real space is marked with notifications and alerts.

In order to implement the new application-based services set forth above, the following functional elements are provided according to the present invention: Context update, Feature selection, and Feature execution, each of which is described in detail below.

Brief Description of the Drawings

A detailed description of the invention is set forth herein below, with reference to the following drawings, in which:

Figure 1 is a functional diagram of the Internet telephony system according to the present invention, which embodies a distributed call processing model;

Figure 2 is a block diagram of a hardware implementation of the Internet telephony system according to a preferred embodiment of the invention;

Figure 3 shows the system architecture and module interactions for the Internet telephony system of Figures 1 and 2;

Figure 4 is a class diagram for the server modules of Figure 3;

Figure 5 is a class diagram for the client modules of Figure 3;

Figure 6 is a use case diagram of the system according to the present invention with corresponding actors;

Figure 7 is a state chart diagram for user login and registration to the system according to the present invention;

Figure 8 is a server welcoming window displayed to the user during operation of the system according to the present invention;

Figure 9 is the server main window displayed to the user during operation of the system according to the present invention;

Figure 10 is an administrator login window displayed to an administrator during operation of the system according to the present invention;

Figure 11 is a context setting window displayed to the administrator during operation of the system according to the present invention;

Figure 12 is a computer name and extension number setting window displayed to the administrator during operation of the system according to the present invention;

Figure 13 is a relationship assigning agent window displayed to the administrator during operation of the system according to the present invention;

Figure 14 is a user rule assigning agent window displayed to the administrator during operation of the system according to the present invention;

Figure 15 is a user-rule conflict resolving agent window displayed to the administrator during operation of the system according to the present invention;

Figure 16 is a user login window displayed to the user during operation of the system according to the present invention;

Figure 17 is a user registration window displayed to the user during operation of the system according to the present invention;

Figure 18 is a client main window displayed to the user during operation of the system according to the present invention;

Figure 19 is a relationship setting window displayed to the user during operation of the system according to the present invention;

Figure 20 is a buddy-list setting window displayed to the user during operation of the system according to the present invention;

Figure 21 is a schedule setting window displayed to the user during operation of the system according to the present invention;

Figure 22 is a user rule setting window displayed to the user during operation of the system according to the present invention;

Figure 23 is a window displayed to the user for selecting context when setting a user rule according to the present invention;

Figure 24 is a window displayed to the user for selecting caller condition when setting a user rule according to the present invention;

Figure 25 is a window displayed to the user for selecting an action when setting a user rule according to the present invention;

Figure 26 is a confirmation window displayed to the user when setting a user rule according to the present invention;

Figure 27 is a flowchart showing the steps for creating a user rule, according to Figures 23 to 26;

Figure 28 is a context simulation window displayed to the user during operation of the system according to the present invention;

Figure 29 is a call delivery agent window displayed to the user during operation of the system according to the present invention;

Figure 30 is an incoming call notification window displayed to the user during operation of the system according to the present invention; and

Figure 31 is a system management agent window displayed to the user during operation of the system according to the present invention.

Detailed Description of the Preferred Embodiment

Turning to Figure 1, a functional diagram is provided of the system according to the present invention. In operation, awareness data 1 is received from one or more ubiquitous sensors(not shown), relating to the user's location and activities, and applied to a context engine 3. A variety of technologies are available for tracking the locations of people. Examples of such ubiquitous sensors include the Active Badge System [Roy Want, Andy Hopper, Veronica Falcao, Jonathan Gibbons, "The Active Badge Location System", *ACM Transactions on Information Systems 10(1) 91-102,* 1992], PARCTabs [Norman Adams, Bill N. Schilit, Rich Gold, Michael Tso and Roy Want, "The PARCTAB Mobile Computing System", *Proceedings of the Fourth Workshop on Workstation Operating Systems (WWOS-IV)*, pages 34-39, Napa, CA, October 1993], mobile phones [Peter Duffet-Smith, "High precision CURSOR and digital CURSOR: the real alternatives to GPS", *Proceedings of EURONAV 96 Conference on Vehicle Navigation and Control,* Royal Institute of Navigation, 1996] and ultrasonic devices [Andy Ward, Alan Jones and

Andy Hopper, "A new location technique for the active office", *IEEE Personal Communications 4(5), 43-7,* 1997].

Our co-pending patent application filed on the same date as this application and entitled "Availability and Location Predictor Using Call Processing Indications" describes a method of evidence gathering that surmises (i.e. makes guesses) about user availability from his/her interactions with a PBX system. The interactions are gathered as awareness information and processed by an algorithm into availability information. By using awareness information as evidence, the algorithm makes predictions about availability of the user or makes decisions in call processing. This information is fed back into a tuple space as assertions that indicate high level assessments about the user's state. These assessments are then used by the call processing components described below to make decisions on call handling.

Additional inventive aspects relating to the application of awareness data 1 to context-based communication systems is set forth in our copending applications filed on the same date as this application and entitled "System and method for facilitating communication using presence and communication services" and "Generation of Availability Indicators from Call Control Policies for Presence Enabled Telephony System".

Awareness data (i.e. raw information about the user) that has been processed by context engine 3 into availability information is then applied to a policy engine 5 in the form of assertions of interest to policies within the system, as described in detail below. The policy engine 5 includes a context update block 9 and feature selection policies 11.

In context update block 9, an incoming event (e.g. Invite, etc.) is related to the current context of the user. Every event has associated with it some indicators that are pertinent to the user's call that provide evidence of the relevance, urgency and importance of the call to the user. Such indicators include caller identity, role relationship between caller and called party, group or project membership, location of user, current state of called user, subject of the call, and so on. Some of these evidential indicators are explicit in the call and some can be derived by inference from other indicators (e.g. awareness data, as discussed above). The context update block 9 uses opportunistic reasoning to

develop the required evidence. This evidence is then supplied to the feature selection policies 11 for selecting a feature, discussed in greater detail below. As discussed in our counterpart application entitled "Generation of Availability Indicators from Call Control Policies for Presence Enabled Telephony System", the evidential indicators may be in the form of fuzzy variables. The fuzziness of these variables is used to indicate the degree of credence that the system has in them.

In the context update evidence gathering process performed by block 9, the system occasionally asks the user which feature he/she wishes to have executed. Sending the user options to his/her wireless browser asking for his/her selection is one among several mutually appropriate options for accomplishing this. Moreover, options may be sent to the calling party asking him/her to select one of several acceptable options.

As discussed above, the user sets some of the policies used in the opportunistic reasoning of context update block 9, although many of the policies are implicit in the way people generally interact. Thus, several default policies are set by system administrators using sociological principles. Examples of such default policies include calls from superiors being more important than calls from subordinates, someone alone in an office being more available than someone with several visitors, calls with a subject related to a user's current activities are less intrusive than calls with unrelated subjects, etc.

Once the user context has been updated with call specific information in block 9, the feature to be performed is selected in block 11. Feature selection policies block 11 utilizes the evidence generated previously in block 9 to direct call handling in accordance with policies set by the user governing the personalized call handling he/she expects. These policies indicate how calls should be handled for differing role relationships, time of day, user state, user location etc. According to applicant's counterpart application filed on the same day as this application and entitled "Personalizable and Customizable Feature Execution for IP Telephony Using Operational Semantics and Deontic Task Trees", block 11 uses forward chaining with fuzzy reasoning to generate priorities among all proposed features and relates that to the intrusiveness of the proposed actions. This results in the selection of a single feature as being the most appropriate for call handling given the declared personalized preferences of the user. The feature is then executed, as shown at block 13.

The execution of features at block 13 can be modulated as described in co-pending UK Application No. 9920051.1 (Call Processing by Synchronized Tuple Spaces) to allow external features acting as enterprise constraints to control the execution of the selected feature.

Thus, in an implementation of the invention that extends the capabilities of SIP, higher levels of abstraction may be specified beyond merely specifying the characteristics of the end point in the ACCEPT-CONTACT and REJECT-CONTACT headers. The end-points may also specify higher level aspects of the user's business relationship. Thus, when an incoming INVITE or other call set up message is received, the content of the message is parsed and used to update the context of the user within block 9 with indicators of the current call. This information is then used to determine which aspect of the user is most appropriate for the call to be directed to. The SIP ACCEPT-CONTACT and REJECT-CONTACT headers are used as evidence in this selection. Appropriate features can then be selected in block 11 based on the derived context, and executed as shown at 13.

With the new addressing capabilities of Internet-based telephony, features can take on new semantics. Instead of remaining at the device level as in conventional telephony, features can operate at a personal level. Calls are not directed to physical endpoints but rather to aspects of the user's identity within his/her business context. A user may have multiple aspects to his/her identity, each of which has different capabilities in the business environment. For example, a user may have multiple aspects to his identity in the form of:

- Voice mail box for messages that do not need current attention
- Secretary or equivalent function that can use mechanisms outside of call processing to make routine decisions about call disposition and other matters to shield the user from interruption
- Identities that represent the user in distinct current projects
- Etc.

These multiple identities may be carried directly in URIs that convey meaning in respect to aspects of the user's identity using the standard 'dot' convention. Thus, a user named John Doe may have multiple aspects of his business identity specified as:

- secretary.john_doe@example.com
- personal.john_doe@example.com
- voice-mail.john_doe@example.com
- project_3499.john_doe@example.com

This change in convention effectively creates an entirely new model of how telephony features operate in converged voice and data systems. As indicated above, according to this invention, features operate with an understanding of the current user context and communications are directed to the most appropriate aspect of the user identity. Thus, for example, calling parties can indicate which aspects of the identity they wish to contact and which aspects they do not wish to deal with. In practice this can be enabled in SIP, by allowing a user to specify multiple end points (called User Agents) where he can be contacted. The user invokes the REGISTER method to declare these locations with a registrar that serves his proxy. When a calling party wishes to make a connection with another user, he may specify contacts that he does not wish to make connection with in a REJECT_CONTACT header and give a list of acceptable contacts with relative priorities for each of them in ACCEPT_CONTACT headers. This convention allows the SIP constructs to enable personal level features beyond the current practice of device level features.

An exemplary hardware implementation of the system according to the present invention is shown in Figure 2. The system is based on a client-server architecture. An Active Call Delivery (ACD) client 21 communicates with an ACD server 27 which, in turn communicates with TSpaces server 23 and a MiTAI Gateway server 25. MiTAI Gateway server 25 provides access via a PBS 26 to the PSTN 28. ACD server 27 is a single server or multiple servers having access to the TSpaces server 23. The ACD server 23 consists of user interfaces and a collection of "system agents" discussed in greater detail below. The user interfaces provide an introduction window, a login window for a system administrator, a context window for managing a hierarchy of contexts and a call

simulation window for simulating a phone call. Each system agent contributes to call processing and it has own responsibilities: a Relationship Assigning (RA) agent is responsible for acquiring the relationship between a caller and a receiver, and assigning it to a relevant data field for call processing. A User Rule Assigning (URA) agent is responsible for extracting all of the matching user rules according to the conditions of each rule and the current context, and assigning them to a relevant data field for call processing. A User-rule Conflict Resolving (UCR) agent is responsible for resolving any conflict that might be present in the assigned rules. As indicated above, these agents do not have to be installed on a particular machine, but can be distributed over a network of machines, which have access to the TSpaces server 23.

The ACD client 21 consists of user interfaces and user agents. The user interfaces provide an introduction window, a login window for a registered system user, and a registration window for a new user. Knowledge management is an important part of the user interface on the client system. A user can create or manage personal information such as a buddy list, relationship information, a schedule and a user preference rule. The client server utilizes two types of agents: The Call Delivery (CD) agent and the System Management (SM) agent. The CD agent acknowledges events, which are generated by the Call Monitor, in the TSpaces server 23. The Call Monitor is a direct interface with the MiTAI Gateway 25, and creates an event that is fed into the TSpaces server 23 for starting call processing by the CD agent. Next, the SM agent acknowledges the event from the CD agent, and distributes the call processing to agents on the network. Although each agent has distinct services, both the server and the client have certain common modules as per customary object-oriented design. These common object modules and the other modules are explained below.

Figure 3 shows the system architecture and module interactions for the Internet telephony system of Figures 1 and 2. The user interface 31 consists of windows, forms, menus and buttons for providing user login, registration, user preference rule setting, context simulation, and a display of messages for assisting the user.

The Event Handler subsystem 33 is a monitoring daemon that resides between the user interface 31 and the Application Level subsystem 35. It waits for physical events to arrive from the user interface 31, such as mouse clicks, and directs them to the appropriate

application module. The development tool, Java, provides embedded event handlers, such as ActionListener, for this purpose.

The Application Level 35 is the core of the system. It consists of multiple agents that provide services for a client as well as for a server. All of the system transactions, functionalities, and knowledge management are performed within this subsystem.

As shown in the class diagram of Figure 4, the server modules are divided into three major parts: System Knowledge Management, Agent Services and Call Simulation. The System Knowledge Management module includes a Context Set sub-module to allow an authorized administrator to create or modify a context hierarchy such as location and activity. The Agent Services module consists of three distinct agent modules: The Relationship Assigning (RA) agent, the User Rule Assigning (URA) agent, and the User-rule Conflict Resolving (UCR) agent. In order to give flexible implementation of the agents, status of the agents is managed in order to know their availability. Network connectivity may affect their availability. Therefore, both an agent and the connection between the agent and the system's TSpaces 37 (Figure 3) must be functional in order to use the agent. The system acquires agent's status by examining a corresponding status tuple in the TSpaces 37. The status tuple consists of "name", "priority" and "availability" fields. Each agent is responsible for updating its status tuple in the TSpaces 37. The procedures for updating a status tuple consist of taking the status tuple and rewriting it with new status information once per second. A tuple can be set to expire after a predetermined time. Upon expiration, the TSpaces server 23 removes the tuple from the TSpaces 37. The expiration time for a status tuple is three seconds, so if an agent fails to update the tuple three times consecutively for any reason, there will be no status tuple for the corresponding agent in the TSpaces 37. The system assumes that an agent is abnormal if there is no status tuple for the agent, or if the "availability" field in the status tuple is set as "abnormal". The time gap between the one-second it takes to update the status tuple and the three seconds allowed before the status tuple expires may prevent unnecessary status toggling by temporal network disturbance.

Each agent is also responsible for registering an event into the TSpaces 37 to communicate with client machines. Whenever an awaited tuple is written into the TSpaces, the TSpaces server 23 notifies this to an agent that registered the event.

Generating an event and getting notification of the event from the TSpaces 37 forms a two-way communication-acknowledgement between agents.

The Relationship Assigning (RA) agent is responsible for responding to a relationship-assigning request from a client's SM agent. The request from a SM agent contains caller and receiver information. The RA agent assigns the relationship between the user and the caller according to the user's buddy-list.

The User Rule Assigning (URA) agent is responsible for responding to a user-rule-assigning request from a client's SM agent. Upon request, the URA agent retrieves both the relationship information and the user's current contexts. The relationship information is a relationship between the caller and the receiver, set by the RA agent. The user's current contexts are the user's location, the current time with the user's schedule, and the user's activity.

- Who is calling?
- Where is the user?
- What the user is doing?
- When is it?

The User-rule Conflict Resolving (UCR) agent is responsible for responding to a client's SM agent for the user-rule conflict-resolving request. The request contains user rule information that is assigned by the URA agent. The UCR agent selects one rule that is the most specific among the assigned rules. The more conditions a rule has, the more specific a rule is considered to be. The Call Simulation service is provided for testing without connecting to a MiTAI Gateway 25. A window with a form serves this functionality.

As shown in the client module class diagram of Figure 5, the client modules are divided into three subsystems: User Knowledge Management, Agent Services, and Context Simulation. A user can manipulate personal knowledge through the User Knowledge Management module.

The Call Delivery (CD) agent is responsible for communication with the phone switch or PBX 26 through MiTAI Gateway 25. In particular, the CD agent registers events to MiTAI Gateway 25 and waits for the notification of an incoming call for a user. When the notification arrives, the CD agent sends a request to the SM agent for further processing and waits for response. This response from the SM agent contains an action to be taken as a result of an entire call processing. Then the CD agent is responsible for requesting the selected action to the MiTAI Gateway 25.

The System Management (SM) agent is responsible for managing other agents' status and sequencing call processing according to the system agents' priority. When the CD agent requests call processing, the SM agent scans the agents' status tuples in the TSpaces 37 and makes a sequence table according to their priority. It sends a processing request to the highest-priority-agent, waits for a response and then sends it to the next-highest-priority-agent. When the SM agent receives a response from the lowest-priority-agent, it sends an information tuple back to the CD agent.

The Context Simulation module serves as a context agent that dynamically detects, interprets and updates the user's current contexts. A Context Simulation window includes all of the possible contexts, which are set by a system administrator, and a user selects from them.

TSpaces 37 (i.e. tuple spaces), are implemented in one or more servers 23 as a network communication buffer with database capabilities. A more complete description of TSpaces 37 may be found in http://www.almaden.ibm.com/cs/TSpaces/, and in co-pending UK Patent Application No. 9920051.1. TSpaces 37 enable communication between applications and devices in a network of heterogeneous computers and operating systems. TSpaces 37 provides group communication services, database services, URL-based file transfer services, and event notification services. The TSpaces 37 are implemented in the Java programming language and thus automatically possess network ubiquity through platform independence, as well as a standard type of representation for all data types. The TSpaces system is appropriate for any application that has distribution

or data storage requirements. It can perform many of the duties of a relational database system without imposing an overly restrictive (and primitive) type system, a rigid schema, a clumsy user interface or a severe runtime memory requirement. In the present invention, the TSpaces Server 23 is the medium between the system and user knowledge stores. It is responsible for managing all tuple-based knowledge transactions, which include read, write, update, take and scan. It is also responsible for event handling, such as registration and notification of events.

The MiTAI Gateway 25 facilitates communication to the MITEL telephony servers (i.e. PBX 26) for processes that are not based on the "C" development language. A description of the MiTAI Gateway 25 may be found in Ramiro Liscano, "MiTAI Gateway", *MITEL Networks*, June 2001. The MiTAI Gateway 25 is a Windows based process that can be executed on any Windows platform. It can manage a single socket connection from any other process on a network and it supports a limited session protocol.

The MiTAI Gateway Server 25 is the intermediate system between the PBX 26 and the ACD's Application Level subsystem 35. The Application Level subsystem 35 registers an event to the MiTAI Gateway server 25 for the purpose of monitoring incoming calls.

In terms of system topology, the User Interface 31 is built on a Windows Platform, and it interacts with the Application Level 35 through an Event Handler 33. The Application Level 35 subsystems use TSpaces server 23 as the medium for communication and for accessing the System Knowledge Management of the server and clients.

All of the knowledge, which includes user information, user rules, the user's current context information, and call information, is stored in the TSpaces 37. The unit of storage is a tuple, as set forth in the references cited above, and as discussed in the Appendix to this specification.

User Information consists of basic user information, relationship information, a buddy list, user preference rules and the dynamically varying user's current context

information. User Information is stored in a tuple named "UserProfile" and the structure is as follows:

{"UserProfile", id, user-info, relationship, buddy-list, user-rule, context}

where "UserProfile"is the name of the tuple, id is user identification for uniquely identifying a user in the system. The user-info field contains basic user information about the user such as: password, name, phone numbers, and the user's time schedule. The phone numbers are extension phone numbers within the PBX 26, such as my office and secretary's phone numbers (e.g. 4001). This field also contains the schedules of the user. Schedules for lunches and meetings may be input by the user directly or ascertained from another application (e.g. Microsoft Outlook, etc.). The relationship field contains relationship information which is defined by the user in a relationship hierarchy using the user interface. The user can add any person as his "buddy" in a buddy list. The buddy list contains information about such persons including as name and phone number, as well as their relationship to the user. The user-rule field contains a user preference rule. The user creates his/her personal preferences for handling incoming calls via the user interface 31. The conditions in a rule can make use of the contexts, the buddy list and a relationship selected from the relationship information hierarchy. In that regard, the context field contains context information. The context determining parameters, which are used in the system, are location, the user's current activity and the present time. The location and activity contexts have a hierarchy, so that they may have sub-contexts. The current context information of a user can be either a real context or a pretended context set by the user. The real context information is updated by the context agent(s) whereas the pretended context, on the other hand, is set and controlled by the user. The pretended context is designed to override the real contexts if so desired by the user. The hierarchy of the location parameters is defined by a system administrator. Thus, if a location's properties are coupled with the phone number, the system can deliver the user's call to the phone nearest to the user's current location.

There are two kinds of activities that can be defined. Some activities can be automatically detected by the system, whereas others can only be assumed or set by a user. For example, the system is able to know whether the user is 'on the phone', but it is difficult to judge if the user is 'busy at work' or 'having a break'. Therefore, detectable

activities are updated by the system automatically, and others are set by the user. A receiver's time context is set according to his time schedule. For example, if the user's lunchtime is scheduled from 12 p.m. to 1 p.m., the system can assume that the user is having lunch during that time period.

Call information is contained in a tuple that an agent shares to communicate with other agents for processing an incoming call. Therefore, it contains all of the necessary data fields for caller information and user preference rules. Agents take the "Call" tuple from the TSpaces 37 and update it according to their responsibility. For example, the RA agent assigns relationships between the caller and the receiver, the URA agent assigns all the appropriate user rules, and the UCR agent resolves user rule conflict by selecting only one user rule. The form of this tuple is:

{"Call", dest-agent, source-agent, id, call-info, user-rule}

where *"Call"* is the name of the tuple, *dest-agent* is the destination agent that is expected to receive this tuple, the *source-agent* field identifies the source agent that sends this tuple, the *id* field is user identification, and the *call-info* field contains basic information of both the caller and the receiver such as phone number, name and relationship information between them. The *user-rule* matches user rule(s) assigned by the agents. The *"Call"* field and the *dest-agent* field are used when agents register an event in the TSpaces server 23. The following is a part of the SM agent event registration routine to the TSpaces server 23:

```
Tuple template = new Tuple("Call", "SMAgent", new Field(String.class),
                           id, new Field(String.class), new Field(String.class),
                           new Field(String.class));
seqNum = ts.eventRegister(TupleSpace.WRITE, template, this, newThread);
```

It requests the TSpaces server 23 to notify the SM agent when a tuple is posted where the first field is *"Call"*, the second is *"SMAgent"*, and the fourth is user *id,* and where the third field is'*new Field(String.class)*' which means that any value will be acceptable for this field.

Module interactions show the behavior of the classes, the modules, and the system as a whole. They describe how the components of the system interact by message passing, function calls, and by sharing the state information. The components interaction for the present invention are shown in the use case diagram and a state chart diagram of Figures 6 and 7, respectively, using Unified Modeling Language (UML) notation.

In order to use the system, including the server system for administrators and the client system for users, a person must be authorized. First time users register by clicking the "Register" button in the "Log-in Window", as indicated in Figure 7. Registering users provides crucial information for using the system such as user-ID, password, name and phone numbers. Every field is filled in without omission before clicking the "OK" button for submitting. Once it is submitted, the system checks validation, such as whether each field has the correct length and is meaningful. The user-ID should be less than 10 alphabetic characters; the password should be less than 10 numbers and/or letters. The name field should be less than 20 characters and phone number fields allow only numbers. If the "Validation Check" stage in Figure 7 is successful, the system writes information to the TSpaces 37 by executing the "write()" operation. The user registration processes are finished when the system successfully writes the user's information into the TSpaces.

Registered users and administrators need to be authenticated before using the system. The fields for user ID and password in the "Log-in Window" have to be correctly filled in, and then the "OK" button clicked. If both fields are filled without omission, the system checks validation of each field. This validation procedure is the same as it for the user registration. The validated user-ID and password pair should be matched with those

in the Tspaces 37. The system obtains the information by executing the "read()" operation and comparing them. The login processes are finished when either the user clicks the "Exit" button or there is a match between the input user ID and password pair and the pair already in the Tspaces 37.

A prototype of the ACD system of the present invention has been implemented using the Java programming language on a Windows NT platform, with the following packages used for the implementation:

- The Java 2 Platform, Standard Edition v1.3.1, for a Java development environment
- The TSpaces v2.1.2 as a data repository and a communication medium between agents
- Mitel Telephony Application Interface (MiTAI) for the PBX interface

The details of the system's installation and execution method include unpacking the Java class files and executing them, as well as other server initiation processes which would be well known to a person of skill in the art.

The design of the ACD system is not restricted to any specific domain of users. A flexible method of defining knowledge for a user domain allows the system to be used in different domains. A system administrator can define a hierarchy of the user's location, activity and time according to a domain of target users. For the purposes of the successful prototype of this invention, the system provides two example domains: an office worker's domain and a professor's domain. A user may choose one of the two domains by clicking the corresponding button in the introduction window, as shown in Figure 8. It sets up the necessary knowledge such as the hierarchy of possible locations, the relationship information and the buddy-list automatically.

The ACD server system 27 is designed to be simple and easy to use. Having connected to the TSpaces Server 23, the installation procedure for the ACD server system

27 requires unpacking the Java class files and executing them on any machine on the network. At the start, a welcoming window provides brief information about the ACD system, administrator login information, and two buttons to load information for testing purposes: "An Office Worker Context Set Loading", and "A Professor Context Set Loading", as shown in Figure 8. When "An Office Worker Context Set Loading" button is clicked, example contexts for an office worker are written into the Tspaces 37. This model of the hierarchy of location and activity is shown in Figure 9. For testing an example of the professor's domain, "A Professor Context Set Loading" can be selected. A tester can start the server without selecting a pre-defined set of information for testing a customized context. The server system informs the tester that a hierarchy of context should be either selected from the two choices or set manually when a tester skips information loading. "ADMIN LOGIN" and "EXIT" buttons are self-explanatory.

To login as an administrator, who controls knowledge and services for the server, the user is authenticated through the login window for an administrator as shown in Figure 10. An error message window is presented if a field is omitted or there is a mismatch between Admin ID and Password.

Once the login is authorized, the Server Main Window is presented for further processing, as shown in Figure 9. In establishing System Knowledge Management (Figure 3), context setting must first be performed to construct a context hierarchy before a client system provides user services. Clicking the "Set" button of a "Context" (Figure 9) allows the administrator to set a hierarchy of contexts using the GUI. The predefined roots of the context hierarchy for this system are *location* and *activity*. Time is another context used in this system, but it is personalized based on a particular user's schedule. Therefore, each client system manages it's own time context. The context window, with an example location hierarchy and activity, is shown in Figure 11. To add a new sub-context, the administrator clicks one of contexts in the hierarchy and clicks the "Add" button. A new "child" context is thereby added with the default name "New Node *n*". Clicking twice on

the name renames the context. To remove a node, the administrator clicks the node to be removed and clicks the "Remove" button. Clicking the "Clear" button clears all the nodes from the context tree. To save changes and finish modifying, the administrator clicks "Finish".

Returning to Figure 9, a computer name and a phone number are paired and saved in order to forward an incoming call. In ACD system 27, this information is used when the delivery action of matching user preference rule is "Forward it to where I am" or when the user wants to forward an incoming call to a different phone. An administrator can add, remove and change the information by clicking the "Set" button for "Comp-ext", which results in display of the table shown in Figure 12.

The server agents can be executed on any machine that has access to the TSpaces server 23. This means that any machine within the network can be used to execute a server agent. This design gives a flexible distribution of agents. All the agents can be executed together on a given machine by clicking the "ALL" button (Figure 9), or each agent can be executed separately either on the same machine or on different machines within the network by clicking the corresponding button. Due to network constraints, each agent should report its status regularly by writing its status tuple once per second, where the lifetime of the tuple is three seconds. The details of status management for the server agents are set forth in detail above with reference to Figures 4 and 5. Each agent has a display window and four buttons to control it, as shown in Figure 13, 14 and 15. Clicking the "Start" button starts a corresponding agent by activate its status report. The "Stop" button is for de-activating its status report for testing purpose. A maximum of three seconds later, the status tuple for corresponding agent no longer exists in the Tspaces 37 with the result that the clients recognize that the agent is not available. The "Start" and "Stop" buttons are exclusive, in that one is disabled when the other is executing. The "Clear" button clears the display area, and the "Finish" button terminates the corresponding agent.

The Relationship Assigning (RA) agent assigns relationship information between the caller and the receiver based on the user's buddy-list. An example of execution is shown in Figure 13, wherein a relationship-assigning request from a System Management (SM) agent for a user, who has a user ID "choi", is received. This request comes with the caller's phone number, which is "4021" in this example. The RA agent gets the user's buddy-list from the TSpaces 37 and finds a relationship between the user "choi" and a person who has the phone number "4021". As a result, a "boss" relationship is found. The call control is sent back to the client by writing a tuple with relationship information into the Tspaces 37. This agent may return multiple relationships. For example, a person may be both a friend and a customer. Tuples for both of these relationships will therefore be returned.

The User Rule Assigning (URA) agent assigns all user preference rules that match the conditions of the rules and the user's current context, as shown in Figure 14. If the condition of a rule is made of information that has a hierarchy, sub-categories are examined. For example, a location condition of a user preference rule is 'If I am in the *office'*. Sub-locations of the *office* such as *lab*, *meeting room* also satisfy the rule's condition. For example, consider the user "choi" receiving a call from extension number "4021" while he is in the *meeting room*, and he is *busy*. In the testing scenario, "4021" is *Thomas Ragan*'s phone and he is the user's *boss*. The matching user preference rules based on his current context, relationship information and the caller are as follows:

Rule name: Worker-office rule
Condition: If a call is coming from [*worker*] relationship
And when I am in my [*office*]
Action: Put through the call

Rule name: Thomas Ragan-busy rule
Condition: If a call is coming from [*Thomas Ragan*]
And if I am [*busy*]
Action: Ask the caller what to do

Rule name: worker-office-busy rule
Condition: if a call is coming from [*worker*] relationship
When I am in my [*office*]
If I'm [*busy*]
Action: Forward it to voice mailbox

The names of the assigning rules are displayed as they are matched in Figure 14. Although these rules are satisfactory for the user's current context, the system needs to select one rule that is most appropriate for the user in order to take an action.

The User-rule Conflict Resolving (UCR) agent selects one user preference rule if there is more than one rule assigned by the URA agent. According to one embodiment of the invention, the UCR selects the most specific among the assigned rules. A rule with more conditions is considered more specific. In the scenario set forth above, the "worker-office-busy rule" is the most specific rule among the assigned rules, and is therefore selected as shown in the Figure 15. However, if a rule has the same number of conditions, the UCR agent looks for a more specific condition by comparing the depth of the condition item in the hierarchy. (e.g. "Meeting Room" is more specific than "Office") When the UCR agent is unable to select one rule among conflict rules by either method set forth above, the system selects the most recently created rule. Specifically, when the UCR agent generates a list of selected rule(s) to the Call Delivery (CD) agent via Tspaces 37, the CD agent assumes there is only one rule assigned by the UCR agent, so it only uses the first rule, which is the most recently created rule (the user rules are saved in order of creation by the user, and the list given to the CD agent is sorted in descending order). Alternatively, the UCR agent may always simply choose a rule at random, or randomly choose a rule in the event of a tie among most specific rules.

As discussed above with reference to Figure 8, when the ACD client starts, the user is presented with a welcoming window. A brief explanation of the project and the testing information are explained on the white text area. Two buttons, the "An office worker info loading" button and the "A professor info loading" button, are used for testing each domain of users. All the necessary personal information for testing the client (user ID, password, user name, phone numbers, hierarchy of personal relationships, the buddy-

list table, the time schedule, and user preference rules) are copied into the TSpaces 37 upon clicking the appropriate button. A confirmation window shows feedback of the result of the processing.

During the login process (Figure 16), the user's ID and password are checked with the information on the TSpaces server 23. If the user is a new system user, registration is effected through the "Register" option. Clicking the "Register" button in the registration window performs validation and verification for each input field. Once a user is logged-in, either through the login window or the registration window, the user name appears on every client window frame as feedback of the user's identification.

A first-time user is prompted to provide basic user information such as user ID, password, name and phone numbers through registration. The input user ID is checked to see if it is not duplicated with existing user IDs. Each field has its own constraints of length and format. An error window notifies the user if any constraint is violated on clicking the "Register" button (Figure 17).

The main window for client control is presented if the login or registration process is successful, as shown in Figure 18. It consists of three parts: User Information, Knowledge Management, and Context Simulation. The basic user information (the user name and the office phone number) is displayed as feedback to the user. The user ID is displayed in the frame of the window. The user can set his personal information such as relationship information, the buddy list, the schedule and the user preference rule through this menu. Each menu has a help button to give a brief explanation of the corresponding item's function.

The personal relationship information is shown as a tree structure that is easy to maintain (Figure 19). To add a new sub-relationship, the user selects one of the relationship nodes and clicks the "Add" button. A new child node is thereby created with a

default name, "New Node *n*", which may be renamed by clicking twice on the name. To remove a relationship, the user selects the removing relationship nodes and clicks the "Remove" button. It should be noted that the sub-relationships that belong to removing relationship are also removed. To remove all of the relationships, the user clicks the "Clear" button to clear all the relationship nodes from the tree. To save changes and finish modifying, the user clicks "Finish".

Clicking any of the fields on the table in Figure 20 allows the user to make changes to that particular field. To remove a set of buddy information from the table, the user selects a column and clicks "Remove". The modified table is saved into the TSpaces 37 when the "Finish" button is clicked.

As shown in Figure 21, a user can set two categorized schedules: lunch time and meeting time. When a user creates a new preference rule, these time settings can be referenced as "lunch time" and "meeting time". The user selects the start time and the end time for each schedule from the pull-down menu in Figure 21. The "Finish" button saves the schedules and removes the Schedule Setting Window.

The user-rule Setting Window consists of three parts: the user rule table, which consists of sequence numbers and user rule names, UI buttons, and a Description window, as shown in Figure 22. Clicking on one of the rules in the table allows the user to see the description of the selected rule in the Description window. The add, refresh, remove, clear and finish buttons are used for managing rules. The "Add" button is designed for creating a new rule and it takes four steps, which are explained in detail below. By clicking the "Refresh" button newly created rules are shown in the user rule table. To remove an existing rule, the user selects a deleting rule on the table and clicks the "Remove" button. To remove all the existing rules, the user clicks the "Clear" button. To finishing editing, the user clicks the "Finish" button to save any changes.

Clicking the "Add" button in the "User-rule Setting Window" starts the creation of a new rule. Adding a new user rule involves four steps. The first step is to select the contexts as a part of the conditions of the rule being created (Figure 23). The location and activity selection are made from a given hierarchy tree. These hierarchies of location and activity are defined by the administrator from a server. The time context is selected from a pull-down menu with three choices: "any time", "meeting time" and "lunch time". The actual time schedules are set by the user through the "Schedule Setting Window." The steps are displayed at the bottom of the window and the current step is written in red. When the context conditions have been selected, the user clicks the "Next" button to move to the second step.

The second step is selecting a type of caller as a part of the condition. One of three categories can be selected: any caller, a buddy list table, and a relationship tree. These three categories are exclusive, so that radio buttons are provided to select only one category. When a category is selected, the user can then select items in its selection window. Figure 24 shows an example of the selection of "Buddy". One of the buddies can now be selected from the buddy table, whereas the relationship hierarchy window remains disabled.

The third step is selecting an action of the rule from the predefined list of actions, as shown in Figure 25. The action items are listed with their associated radio buttons, and only one can be selected from the list.

The fourth and the last step for creating a new rule is confirmation. As shown in Figure 26, the user confirms and assigns a unique rule name. The "Description for the rule" window shows the selections the user made: the condition(s) and an action. Clicking 'Submit' saves the new rule.

The description of the rule as an example for creating a new rule (from figures 23 to 26) is as follows:

Rule name: Mindy Baker-office room-busy rule
Condition: If a call is coming from [*Mindy Baker*]
And when I'm in my [*office room*]
And when I'm [*busy*]
Action: Forward it to secretary

The entire process to create a user preference rule is shown in Figure 27.

Ultimately, the user's current contexts such as the current location and activity are updated by the Context Agent. In the successful prototype, a simulation program was used to substitute the occurrence of events in real-life. For testing purposes, the tester selects one of the desired contexts on the hierarchy trees, and then clicks the "Apply" (Figure 28). The current time, which is displayed on the window, is the client machine's system time that is used as time context by matching with a user's schedule.

As discussed above, the client has two agents: the Call Delivery Agent (CD) agent and the System Management Agent (SM) agent. Each agent has its own display window to present procedural messages to a user. The CD agent is connected to both the TSpaces server 23 to communicate with other agents and to the MiTAI Gateway server 25 to communicate with the phone system.

The window in Figure 29 displays the machine name and the port number of the TSpaces server 23, which this client is connected to. The default TSpaces server name is "localhost", which is the same machine as the current client machine. The second line shows the MiTAI Gateway server name and its port number. The "CDAgent for [choi] is now ready" means that the two necessary connections are confirmed, and the CD agent is ready for the user, whose user ID is "choi".

A call processing request can be received from either the Call Simulation or the Call Monitor. The Call Monitor communicates with the MiTAI Gateway server 25 for handling actual phone calls whereas the Call Simulation is another window on a server machine to test the system without MiTAI Gateway interfaces. When call processing has finished involving all the available agents, the CD agent extracts the selected user rule, which is a result of the processing, and requests the Call Monitor to carry out the action stated in the selected rule. When the example in Figure 29 is executed, the action, "Notify me on the screen", results in a notification window on the client machine, as shown in Figure 30.

The SM agent is also connected to the TSpaces server 23 to communicate with other agents. The display in Figure 31 confirms the established connection. The default TSpaces server name is "localhost", which is the same as the CD agent's default server name. "SMAgent is for [choi] now ready" means that the necessary connection is confirmed and the SM agent is ready for the user, whose user ID is "choi". The SM agent is responsible for sequencing the available agents according to their priority. The display window shows the sequencing of the agents as a part of the call processing. When the CD agent notifies the user about an incoming call, the SM agent retrieves the status of the agents and distributes a call control to the each agent. On the completion of the call processing, the control is sent back to the CD agent to execute the selected action. The SM agent window has an "Agent Status" button that allows the user to check the agent status manually. The "Clear" button clears the message display area and the "Finish" button exits the system.

In summary, according to the present invention a model of contexts is defined for a message delivery system and a system architecture provided that facilitates the creation of context- and rule-based communication. Location context is used to forward incoming calls based on location information. The user's activity or availability is used in the instant

invention to notify other connected users about the user's status such as "busy", "be back", "away" and "lunch". The time context is used to set the time of application of certain user-defined rules.

By receiving the appropriate messages in the right situation, system users benefit from minimal interruptions. By exploiting personal traits and filtering messages based on both a user's current context models and his/her preference rules, the possibility of having desirable delivery action for the user is enhanced. Although the user's current contexts have been simulated for the purpose of the working prototype of the system set forth in detail above, a person of skill in the art will readily appreciate that the system can be implemented with a Context Agent, which actually detects a user's contexts. To that end, a simple type of the Context Agent, which detects a computer's mouse movement, has been tested.In operation, a user who uses multiple machines in the network first logs into a particular computer. The Context Agent detects the computer's mouse movement and in response updates the user's location information in the Tspaces 37 so that an incoming call can be notified or forwarded to the user at that location.

As discussed in our co-pending application entitled "Privacy and Security for Presence Systems with Tuple Spaces", although using the TSpaces 37 provides great flexibility in the aspect of multiple agent system design, it suffers from a weakness in security since it allows for the sharing of all information. Some of the privacy-sensitive information such as the user profile should be protected. The TSpaces server 23 provides access control by setting user and group permissions on the Tspaces 37 so that only users with the proper access control permissions may read and write tuples from the TSpaces. Additional security measures are set forth in the aforenoted copending application.

Also, although the successful prototype has been described in terms of call processing, it is contemplated that the principles of the invention may be extended to implement context-based event handling beyond telephony, such as email processing, visitor notification services, etc.

All such modifications and variations are believed to be within the sphere and scope of the present invention as defined by the claims appended hereto.

The following Appendix sets forth in greater detail the structure and operation of specific tuples used to implement the successful prototype:

Appendix: Structure of Knowledge

Knowledge is stored in the Tspaces 37. The unit of knowledge is a "tuple", which is also the unit of data transaction with the TSpaces. A tuple is an aggregation of field(s), and maximum of seven fields are allowed by the TSpaces. The first field is reserved for the name of a tuple, and other fields can be either a simple string or a string with tags[1]. The simple string will be marked as 'String', and the string with tags will be marked as 'Tag String' in the definitions below. There are four tuples for this system: 'UserProfile', 'Call', 'Context', and 'AgentInfo'.

There are user-defined fields: the *'relationship'* in the 'UserProfile', the *'location'* and the *'activity'* in the 'Context'. The knowledge structure for those fields will be defined during run-time by using the GUIs. Those fields will be marked as the 'User Defined'. Some fields allow saving many same-format-fields. For example, the 'userrule' field in the 'UserProfile' tuple allows saving many user rules in that field. (marked as *)

1. UserProfile

1.1 Description
This is user specific information, includes basic user information, for call delivery.

1.2 Tuple fields

Tuple Name :: 'UserProfile'
+ id : String
+ userinfo : Tag String
+ relationship : Tag String – User Defined
+ buddylist* : Tag String
+ userrule* : Tag String
+ context : Tag String

[1] Each field contains simple, but flexible data. It uses tags similar to XML format, but it does not use any particular parser due to the problems of TSpaces with XML. It uses internal function to have access to data instead of parsing.

- userinfo Field: This field contains fundamental user information for operating the system.

Field Name :: 'userinfo'
+ passwd : String
+ fname : String
+ lname : String
+ office : String
+ home : String
+ cell : String
+ secretary : String
+ lunch time : String
+ meeting time : String

```
<?xml version="1.0"?>[2]
<!DOCTYPE userinfo [
  <!ELEMENT userinfo (passwd, fname, lname, office,
home?, cell?, secretary?, lunch time?, meeting time?)>
  <!ELEMENT passwd  (#PCDATA)>
  <!ELEMENT fname   (#PCDATA)>
  <!ELEMENT lname   (#PCDATA)>
  <!ELEMENT office  (#PCDATA)>
  <!ELEMENT home    (#PCDATA)>
  <!ELEMENT cell    (#PCDATA)>
  <!ELEMENT secretary(#PCDATA)>
  <!ELEMENT lunch time (#PCDATA)>
  <!ELEMENT meeting time (#PCDATA)>
]>
```

- relationship Field: This field contains the hierarchy of personalized relationship information that is defined by a user using GUI.
- buddylist Field: This field contains a list of buddy and their information, which is set by a user

Field Name :: 'buddylist'
+ fname : String
+ lname : String
+ phone : String
+ relationship : String

```
<?xml version="1.0"?>
<!DOCTYPE buddylist [
 <!ELEMENT buddylist(fname,lname,phone,relationship)>
  <!ELEMENT fname   (#PCDATA)>
  <!ELEMENT lname   (#PCDATA)>
  <!ELEMENT phone   (#PCDATA)>
  <!ELEMENT relationship  (#PCDATA)>
]>
```

- userrule Field: This field contains a list of user rules, which are created by a user

Field Name :: 'userrule'
+ rulename : String

[2] XML DTD is for reference only. The implementation did not use these DTDs.

+ condition : String
+ fname : String
+ lname : String
+ phone : String
+ relationship : String
+ location : String
+ activity : String
+ time : String
+ action : String

```
<?xml version="1.0"?>
<!DOCTYPE userrule [
  <!ELEMENT userrule (rulename, condition, action)>
  <!ELEMENT rulename  (#PCDATA)>
  <!ELEMENT    condition(fname?,    lname?,    phone?,
relationship?, location?, activity?, time?)>
  <!ELEMENT fname(#PCDATA)>
  <!ELEMENT lname(#PCDATA)>
  <!ELEMENT phone(#PCDATA)>
  <!ELEMENT relationship(#PCDATA)>
  <!ELEMENT location (#PCDATA)>
  <!ELEMENT activity (#PCDATA)>
  <!ELEMENT time     (#PCDATA)>
  <!ELEMENT action (#PCDATA)>
]>
```

- context field: This field contains the user's current contexts, which are set by either simulation agent or context agent.

Field Name :: 'context'
+ location : String
+ activity : String

```
<?xml version="1.0"?>
<!DOCTYPE context [
  <!ELEMENT context (location, activity)>
  <!ELEMENT location(#PCDATA)>
  <!ELEMENT activity(#PCDATA)>
]>
```

2. Call

2.1 Description

This tuple is for call processing among agents.

2.2 Tuple fields

Tuple Name :: 'Call'
+ destagent : String
+ sourceagent : String
+ id : String
+ callinfo : Tag String
+ userrule* : Tag String

+ policyrule* : Tag String

- destagent Field: This field contains a name of the destination agent that will receive this tuple.
- sourceagent Field: This field contains a name of the source agent that is sending this tuple.
- id Field: This is a unique user id that using as a key value to scan and read tuples.
- callinfo Field: This field contains basic information of both a caller and a receiver such as phone number and name. It also contains relationship information between them.

Field Name :: 'callinfo'
+ callerinfo : String
+ phone : String
+ fname : String
+ lname : String
+ calleeinfo : String
+ phone : String
+ fname : String
+ lname : String
+ relationship : String

```
<?xml version="1.0"?>
<!DOCTYPE callinfo [
  <!ELEMENT    callinfo    (callerinfo,    calleeinfo,
relationship?)>
  <!ELEMENT callerinfo (phone, fname, lname)>
  <!ELEMENT phone (#PCDATA)>
  <!ELEMENT fname (#PCDATA)>
  <!ELEMENT lname (#PCDATA)>
  <!ELEMENT calleeinfo (phone, fname, lname)>
  <!ELEMENT phone (#PCDATA)>
  <!ELEMENT fname (#PCDATA)>
  <!ELEMENT lname (#PCDATA)>
  <!ELEMENT relationship(#PCDATA)>

  ]>
```

- userrule Field: This field contains user preference rules that are selected by agent(s). The format of this field is exactly the same as that 'UserProfile' has.

3. Context

3.1 Description

This is the hierarchy of context information that will be decided and input by system administrator. The contexts, which will be used in our prototype system, are a user's location, his/her current activity and the present time.

3.2 Tuple fields

Tuple Name :: 'Context'
+ location : String
+ activity : String

4. AgentInfo

4.1 Description

Agents are responsible for updating their status into the TSpaces. The SM agent uses this agent information when it decides the sequence of the process according to their availability.

4.2 Tuple fields

Tuple Name :: 'AgentInfo'
+ agentname : String + priority : String + status : String

```
<?xml version="1.0"?>
<!DOCTYPE agentinfo [
  <!ELEMENT agentinfo (agentname,priority,status)>
  <!ELEMENT agentname   (#PCDATA)>
  <!ELEMENT priority(#PCDATA)>
  <!ELEMENT status   (#PCDATA)>
]>
```

Claims:

1. A communication system comprising:

at least one ubiquitous sensor for generating awareness data relating to a user;

a context engine for receiving and processing said awareness data to determine the user's current context for the purpose of event handling;

a policy engine for receiving and relating at least one pertinent evidential indicator of an incoming event from a caller to said user's current context and in response selecting a preferred event handling feature; and

a delivery agent for executing said preferred event handling feature.

2. A communication system as claimed in claim 1, wherein said context engine further comprises an algorithm for processing said awareness data into availability information and applying said availability information to said policy engine as tuple space assertions.

3. A communication system as claimed claim 1 or claim 2, wherein said policy engine further comprises a context update block for developing and expressing said evidential indicator as a fuzzy variable, and a feature selection policies block for receiving said evidential indicator and in response utilizing forward chaining with fuzzy reasoning to generate priorities among a plurality of proposed features and selecting said preferred event handling feature to have the least intrusiveness relative to event handling policies set by said user.

4. A communication system as claimed in any one of claims 1 to 3, wherein said at least one evidential indicator includes at least one of caller identity, role relationship between caller and said user, group or project membership, location of said user, current activity of said user, and subject matter of said event.

5. A communication system as claimed claim 3, wherein said context update block prompts said user to select said preferred event handling feature.

6. A communication system as claimed in claim 3 or claim 5, wherein said feature selection policies block receives additional event handling policies set by an administrator.

7. A communication system as claimed in claim 3 or claim 5, wherein said feature selection policies block receives said event handling policies from said user as ACCEPT-CONTACT and REJECT-CONTACT headers to identify SIP endpoints.

8. A communication system as claimed in any one of claims 1 to 7, wherein said policy engine stores a plurality of unique identifiers for identifying specific aspects of said user's identity that are related to said user's context and selecting a preferred event handling feature based on said aspects to create personalized features.

9. A system as claimed in claim 1 and substantially as hereinbefore described with reference to or as illustrated in the accompanying Figures 1 to 31 inclusive.

Abstract

A communication system comprising one or more ubiquitous sensor(s) for generating awareness data relating to a user, a context engine for receiving and processing the awareness data to determine the user's current context for the purpose of event handling, a policy engine for receiving and relating at least one pertinent evidential indicator of an incoming event from a caller to the user's current context and in response selecting a preferred event handling feature, and a delivery agent for executing the preferred event handling feature.

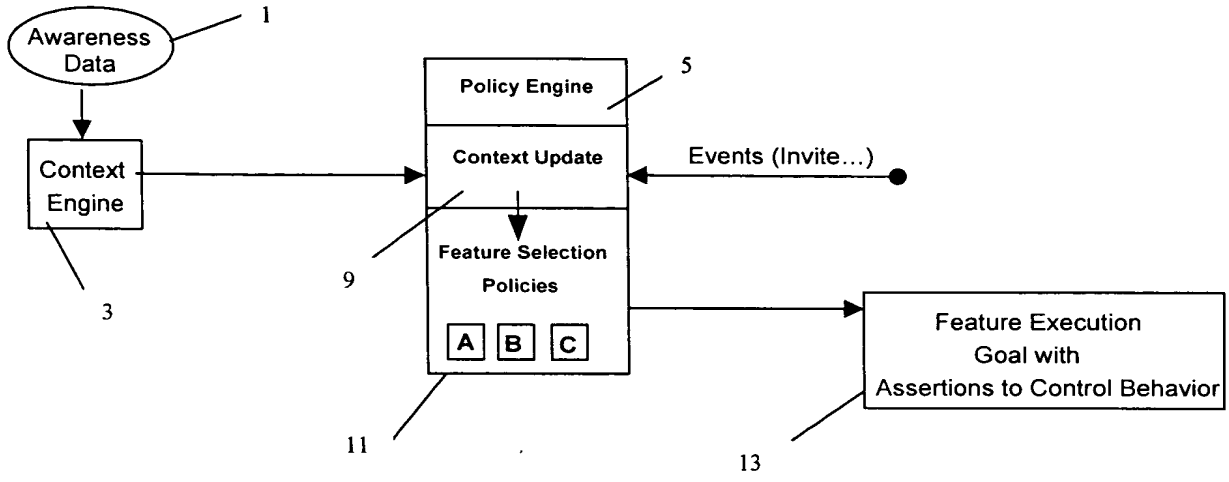

Awareness Data
1
Context Engine
3
Policy Engine
5
Context Update
Events (Invite...)
Feature Selection Policies
A
B
C
9
11
Feature Execution Goal with Assertions to Control Behavior
13

Figure 1

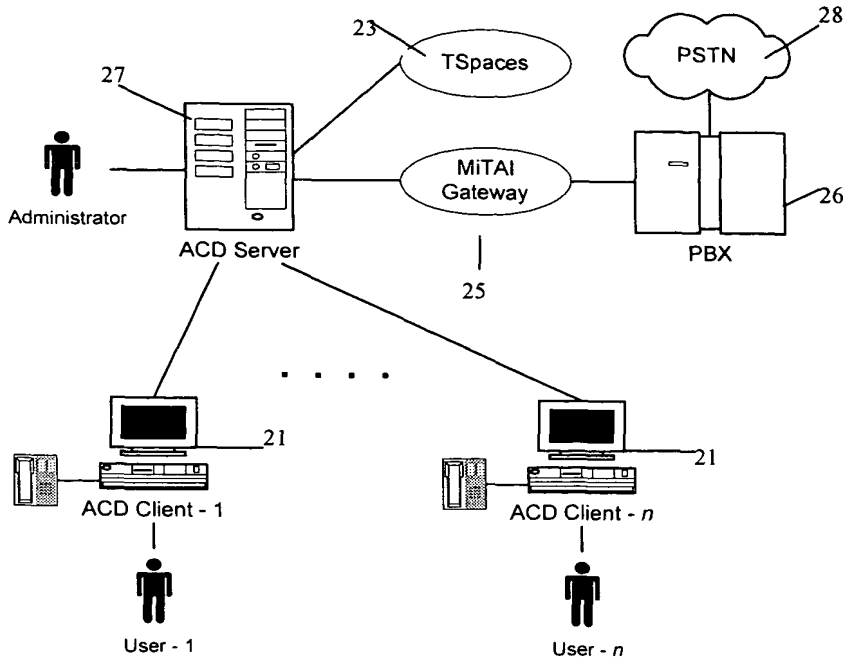


Figure 2

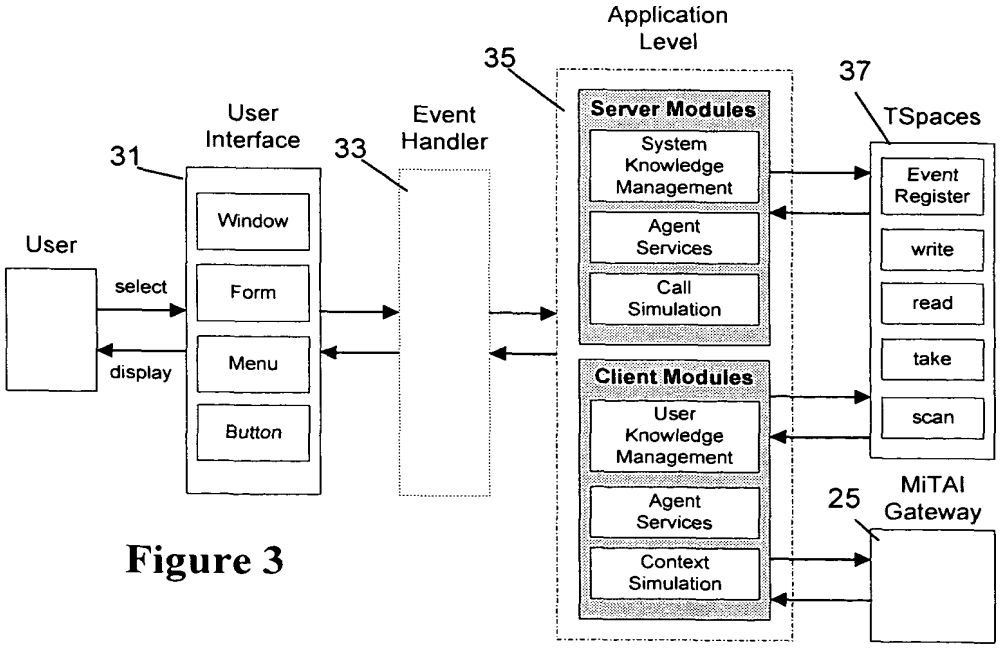


Figure 3

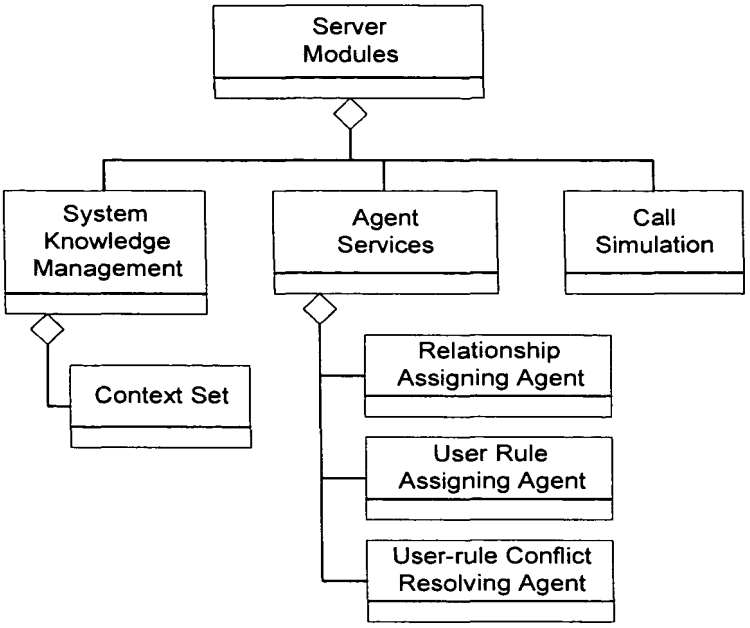


Figure 4

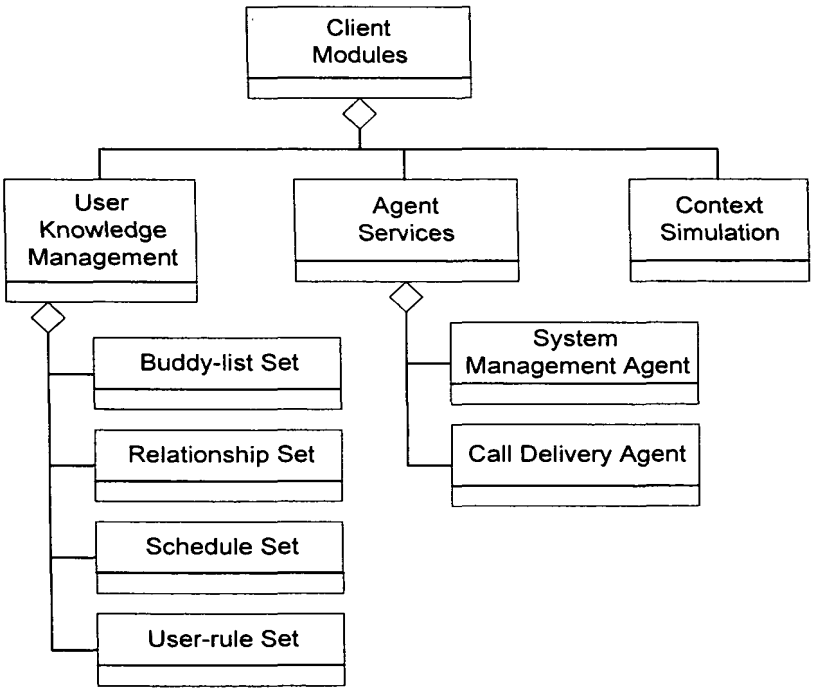


Figure 5

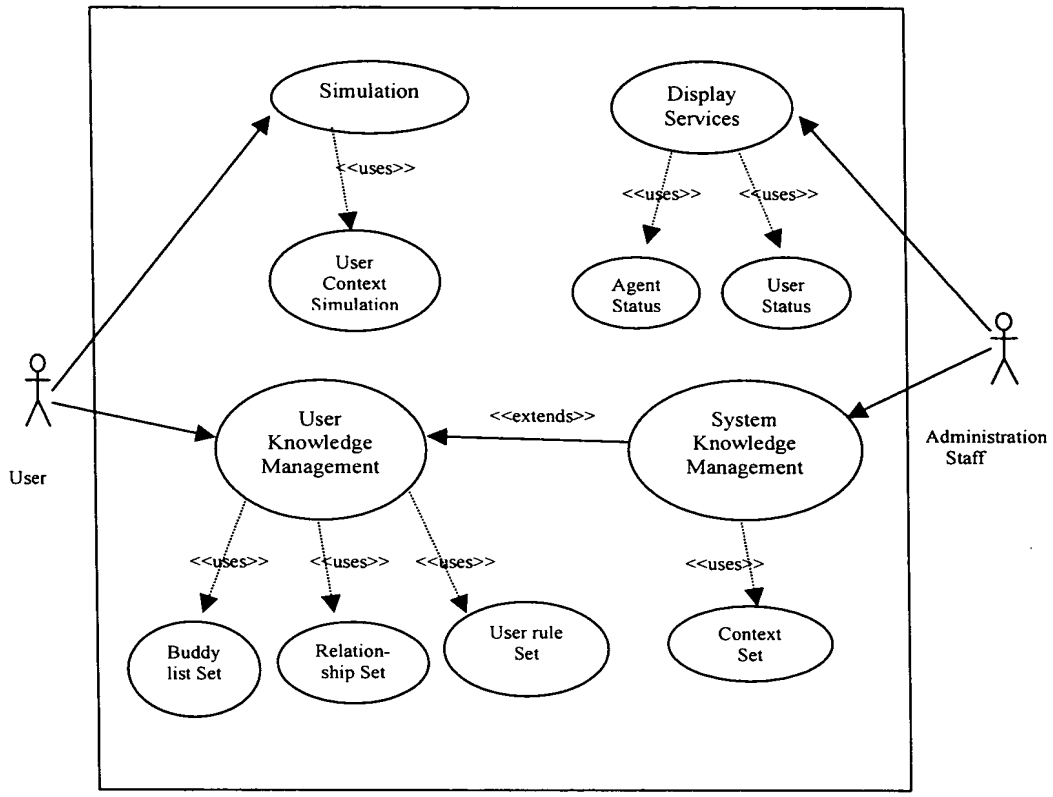

Simulation
<<uses>>
User Context Simulation
Display Services
<<uses>>
<<uses>>
Agent Status
User Status
User
User Knowledge Management
<<extends>>
System Knowledge Management
Administration Staff
<<uses>>
<<uses>>
<<uses>>
<<uses>>
Buddy list Set
Relation-ship Set
User rule Set
Context Set

Figure 6

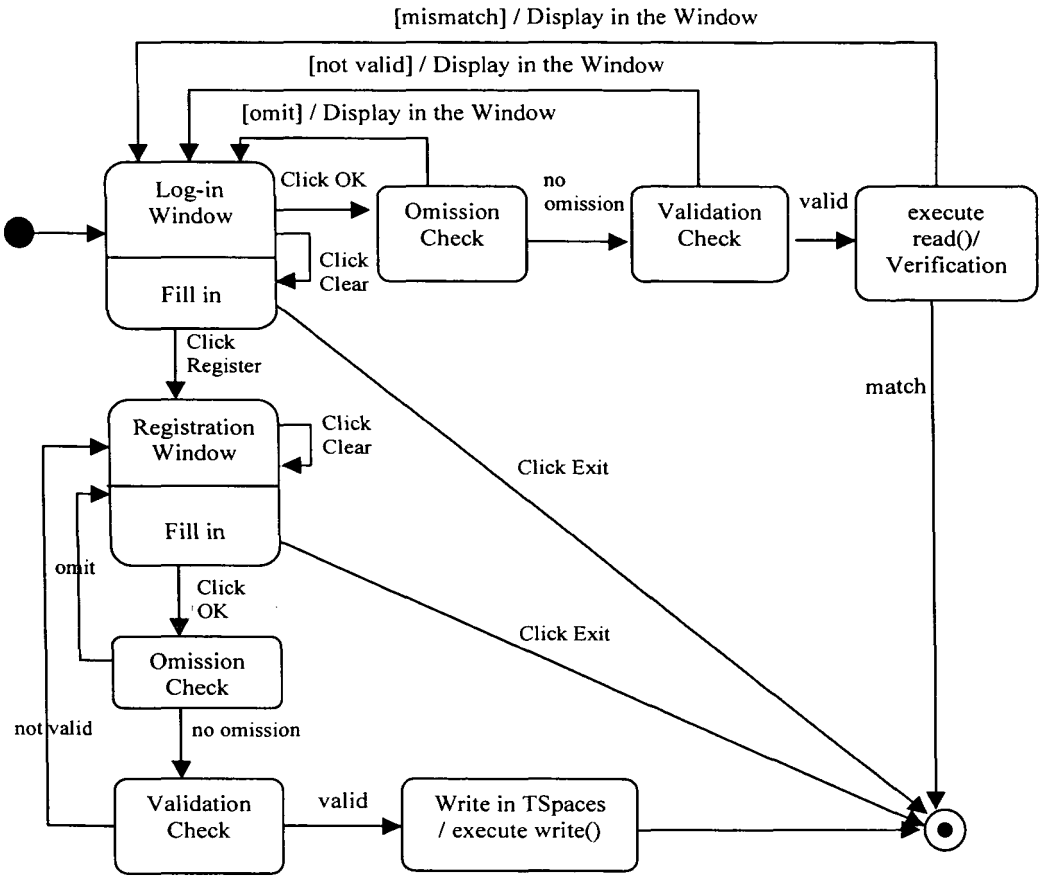

[mismatch] / Display in the Window
[not valid] / Display in the Window
[omit] / Display in the Window
Log-in Window
Fill in
Click OK
Click Clear
Omission Check
no omission
Validation Check
valid
execute read()/ Verification
match
Click Register
Registration Window
Fill in
Click Clear
Click Exit
omit
Click OK
Omission Check
Click Exit
not valid
no omission
Validation Check
valid
Write in TSpaces / execute write()

Figure 7

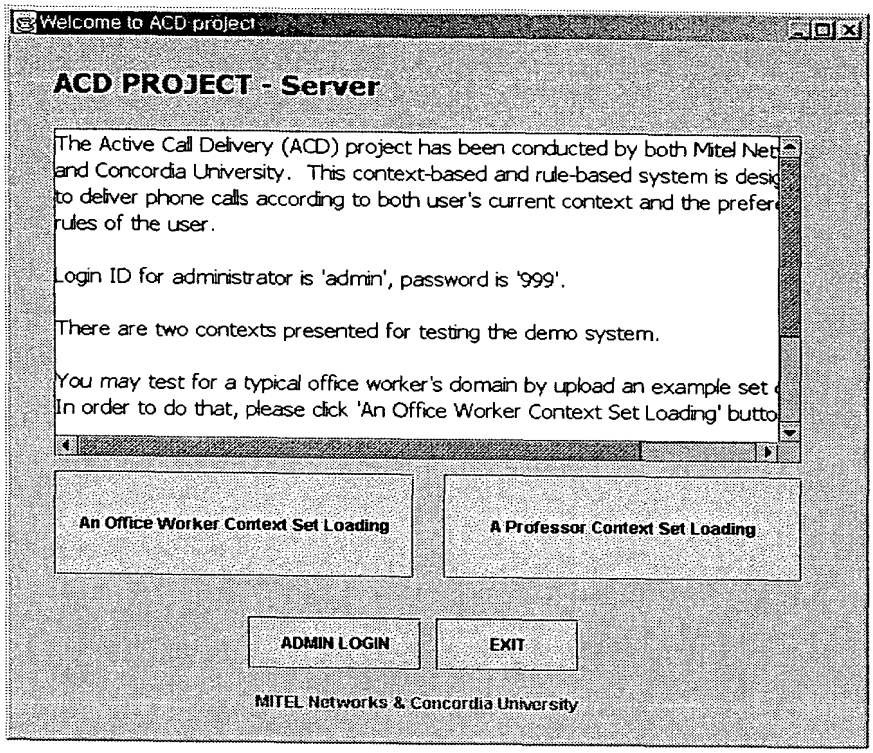

Welcome to ACD project
ACD PROJECT - Server
The Active Call Delivery (ACD) project has been conducted by both Mitel Net
and Concordia University. This context-based and rule-based system is desig
to deliver phone calls according to both user's current context and the prefer
rules of the user.
Login ID for administrator is 'admin', password is '999'.
There are two contexts presented for testing the demo system.
You may test for a typical office worker's domain by upload an example set
In order to do that, please dick 'An Office Worker Context Set Loading' butto
An Office Worker Context Set Loading
A Professor Context Set Loading
ADMIN LOGIN
EXIT
MITEL Networks & Concordia University

Figure 8

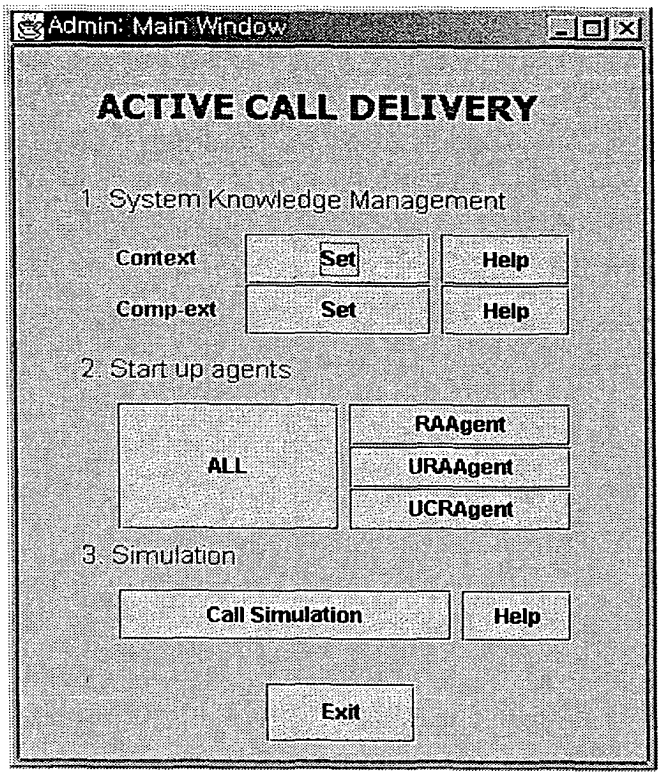

Admin: Main Window
ACTIVE CALL DELIVERY
1. System Knowledge Management
Context
Set
Help
Comp-ext
Set
Help
2. Start up agents
ALL
RAAgent
URAAgent
UCRAgent
3. Simulation
Call Simulation
Help
Exit

Figure 9

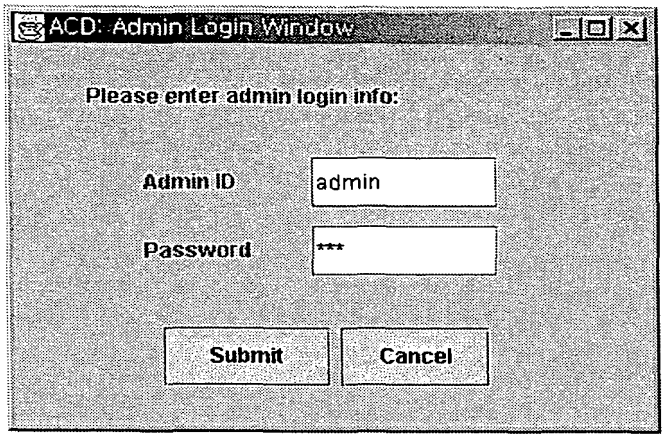


Figure 10

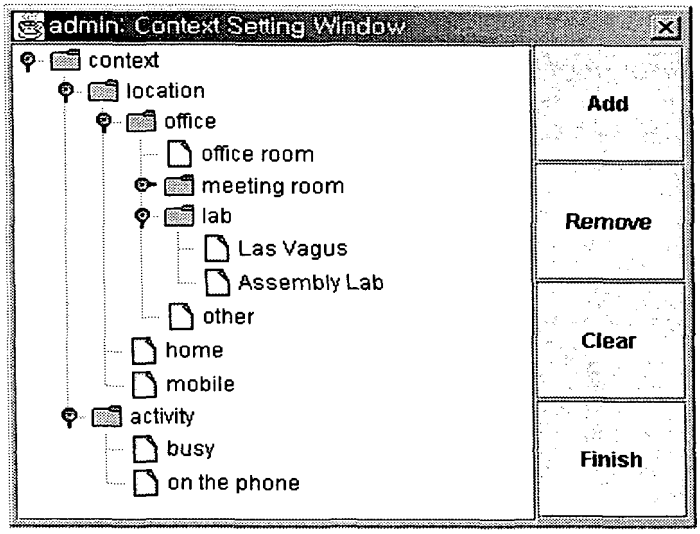


Figure11

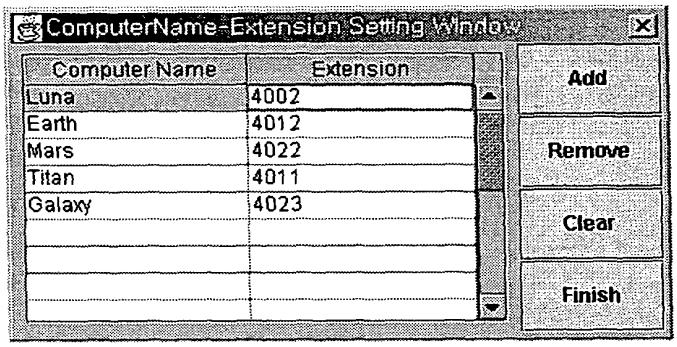


Figure 12

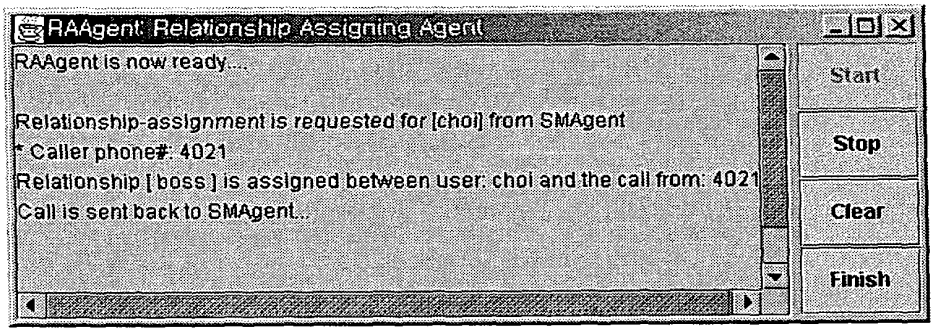


Figure 13

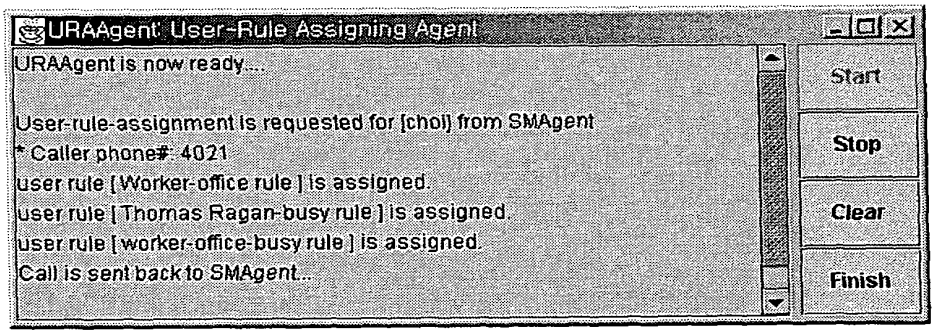


Figure 14

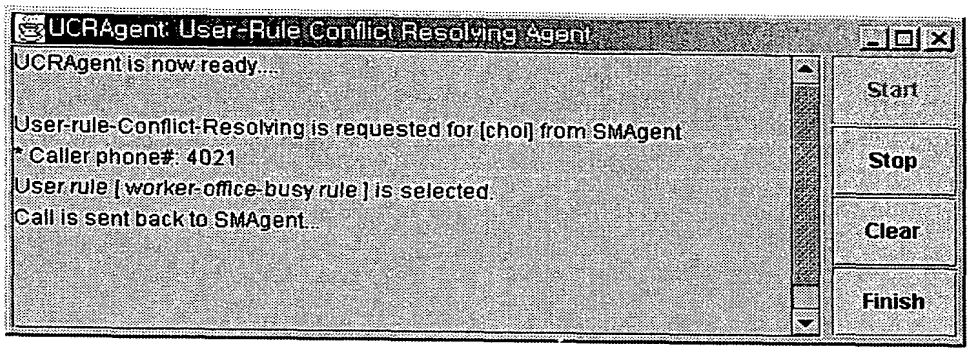


Figure 15

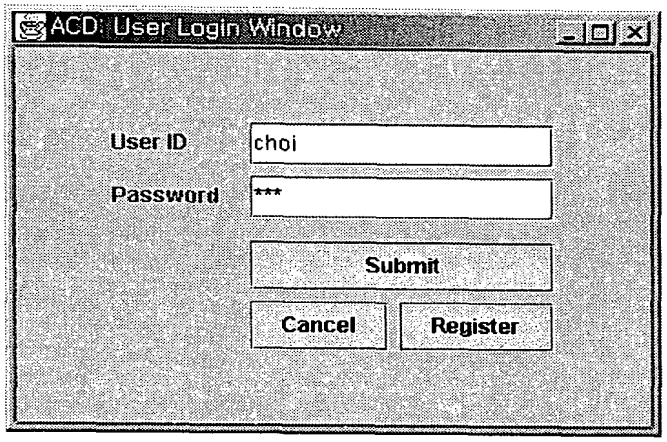
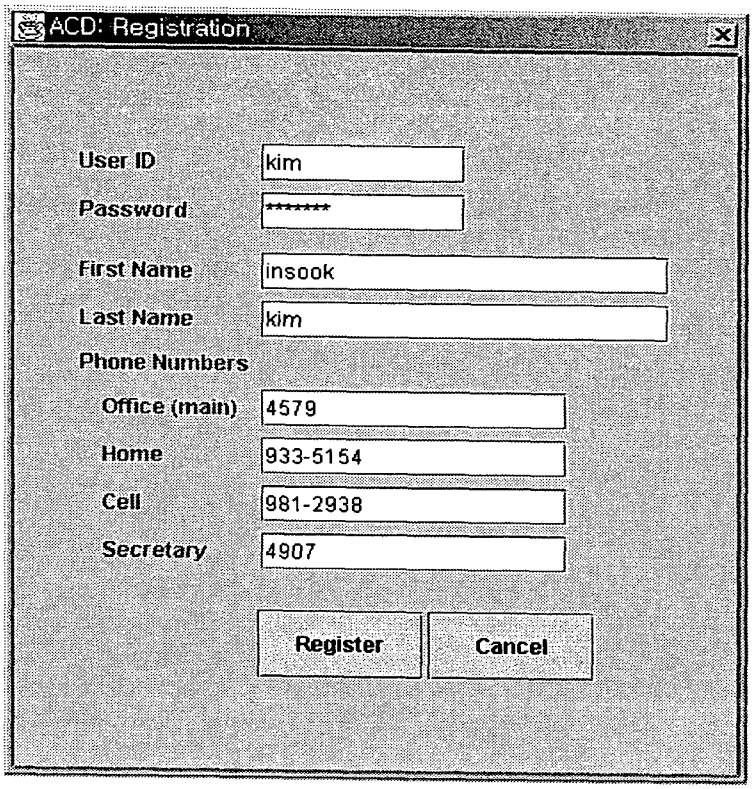


Figure 16


ACD: Registration
User ID
kim
Password

First Name
insook
Last Name
kim
Phone Numbers
Office (main)
4579
Home
933-5154
Cell
981-2938
Secretary
4907
Register
Cancel

Figure 17

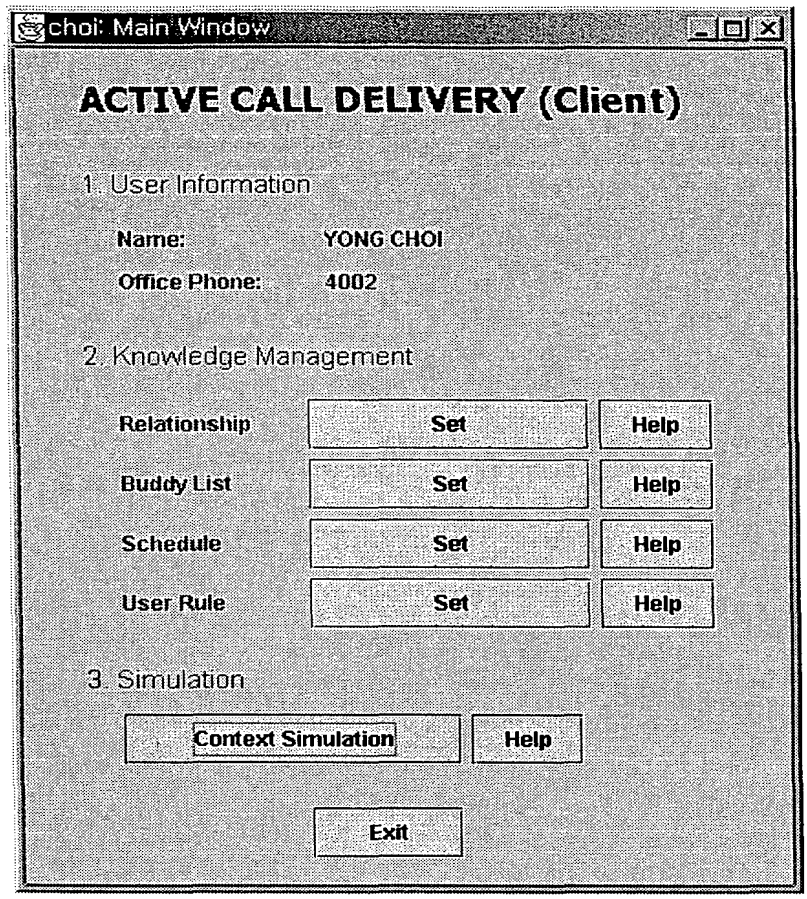

choi: Main Window
ACTIVE CALL DELIVERY (Client)
1. User Information
Name:
YONG CHOI
Office Phone:
4002
2. Knowledge Management
Relationship
Set
Help
Buddy List
Set
Help
Schedule
Set
Help
User Rule
Set
Help
3. Simulation
Context Simulation
Help
Exit

Figure 18

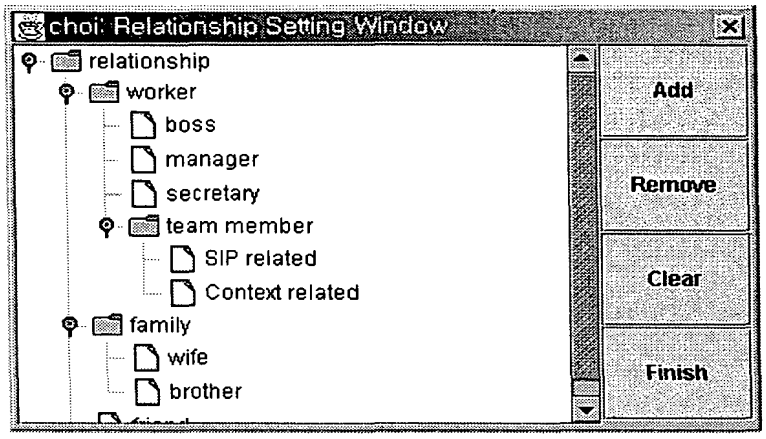


Figure 19

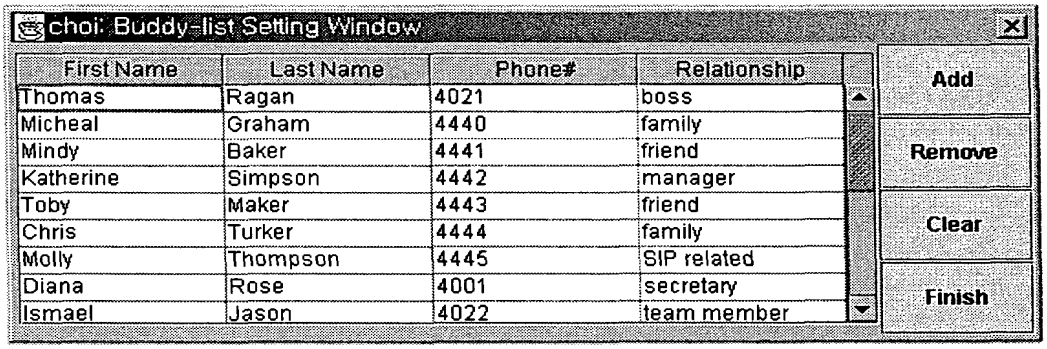


Figure 20

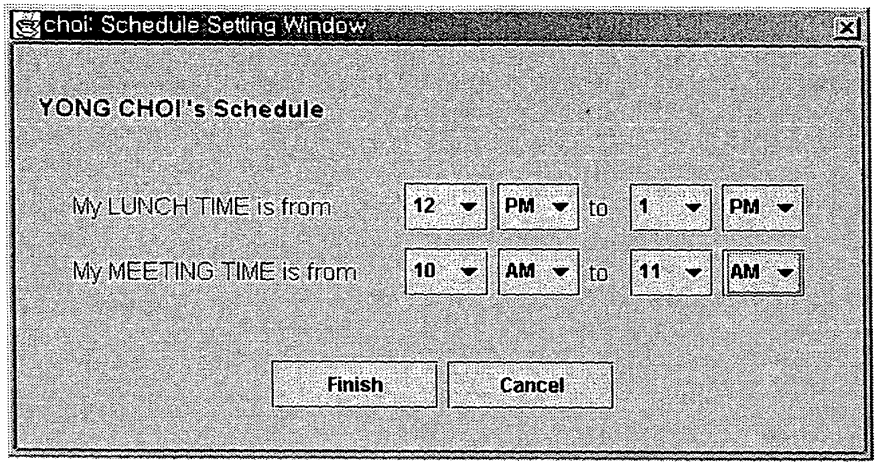


Figure 21

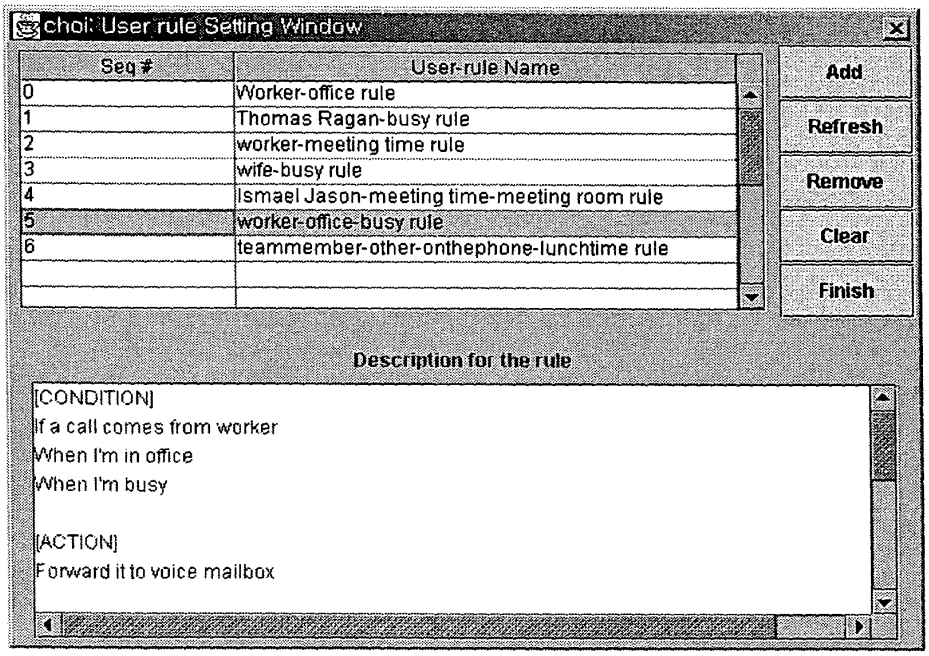


Figure 22

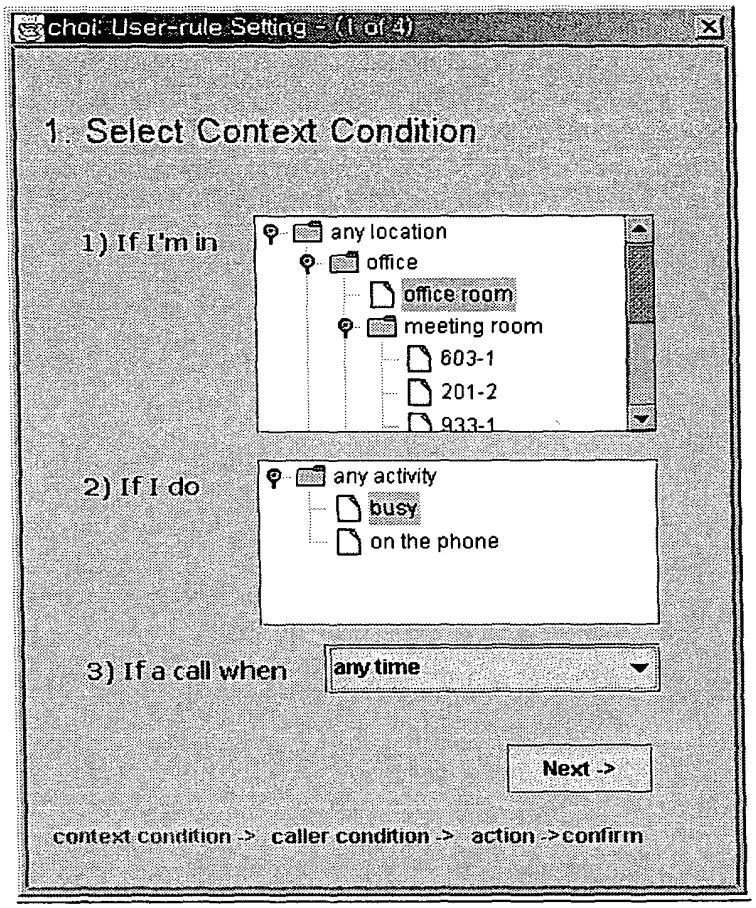

choi: User-rule Setting - (1 of 4)
1. Select Context Condition
1) If I'm in
any location
office
office room
meeting room
603-1
201-2
933-1
2) If I do
any activity
busy
on the phone
3) If a call when
any time
Next ->
context condition -> caller condition -> action ->confirm

Figure 23

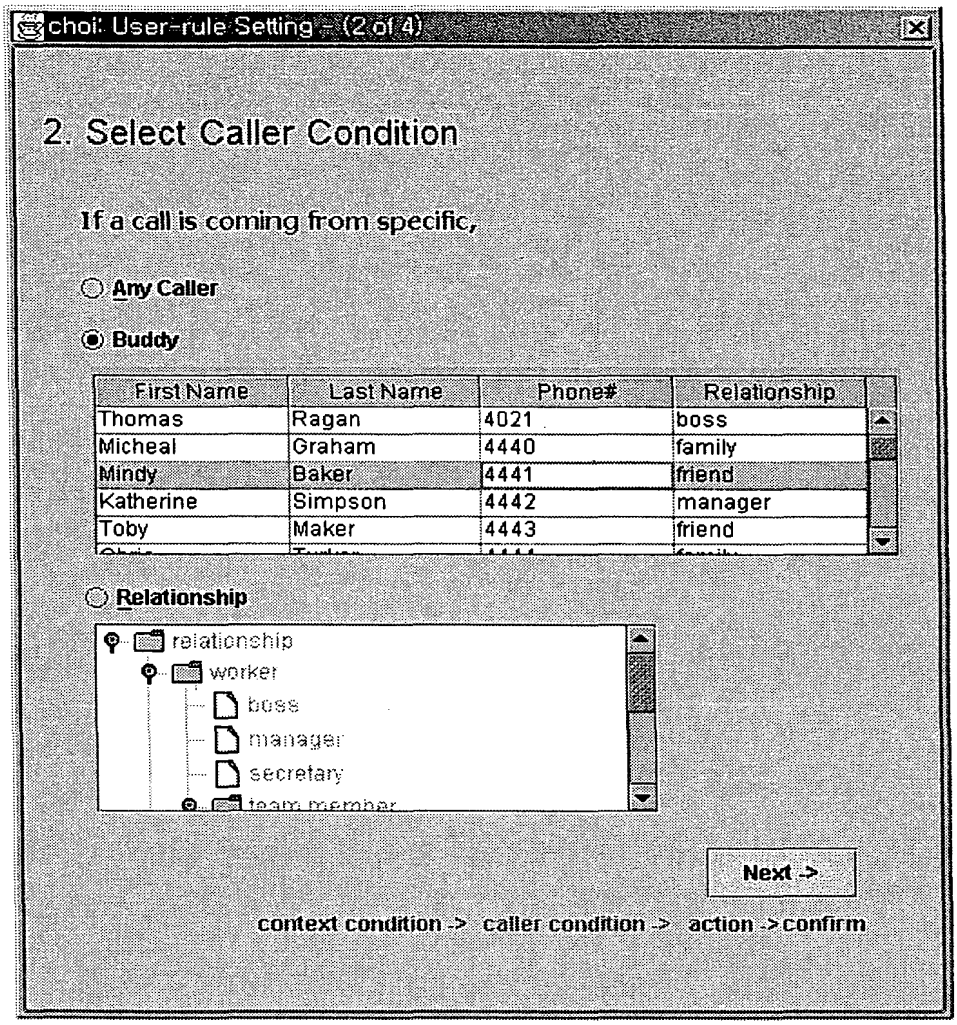

choi: User-rule Setting - (2 of 4)
2. Select Caller Condition
If a call is coming from specific,
Any Caller
Buddy
First Name
Last Name
Phone#
Relationship
Thomas
Ragan
4021
boss
Micheal
Graham
4440
family
Mindy
Baker
4441
friend
Katherine
Simpson
4442
manager
Toby
Maker
4443
friend
Relationship
relationship
worker
boss
manager
secretary
Next ->
context condition -> caller condition -> action ->confirm

Figure 24

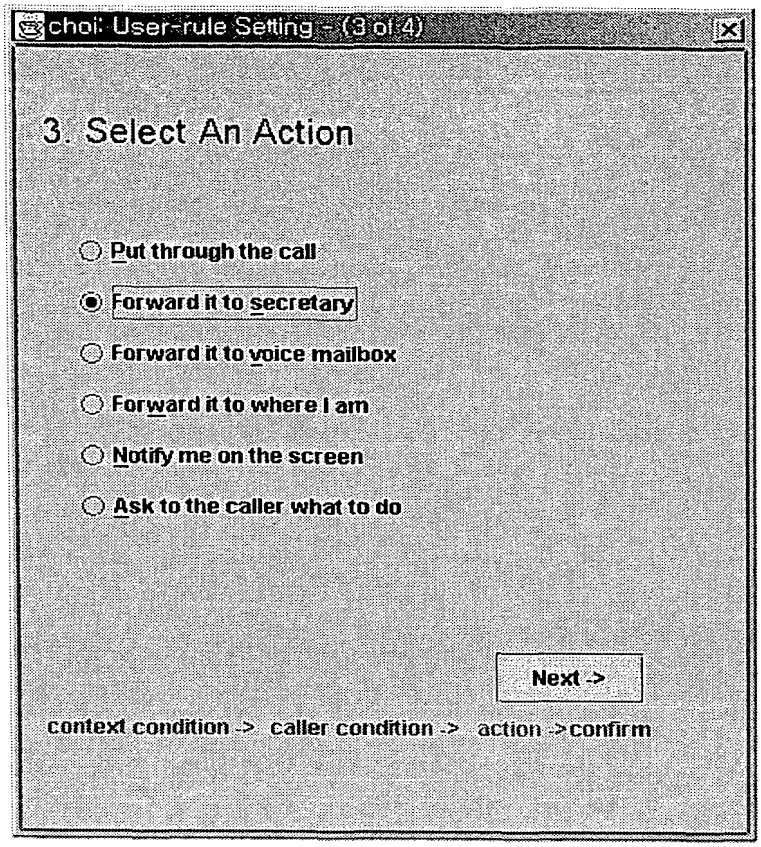

choi: User-rule Setting - (3 of 4)
3. Select An Action
Put through the call
Forward it to secretary
Forward it to voice mailbox
Forward it to where I am
Notify me on the screen
Ask to the caller what to do
Next ->
context condition -> caller condition -> action ->confirm

Figure 25

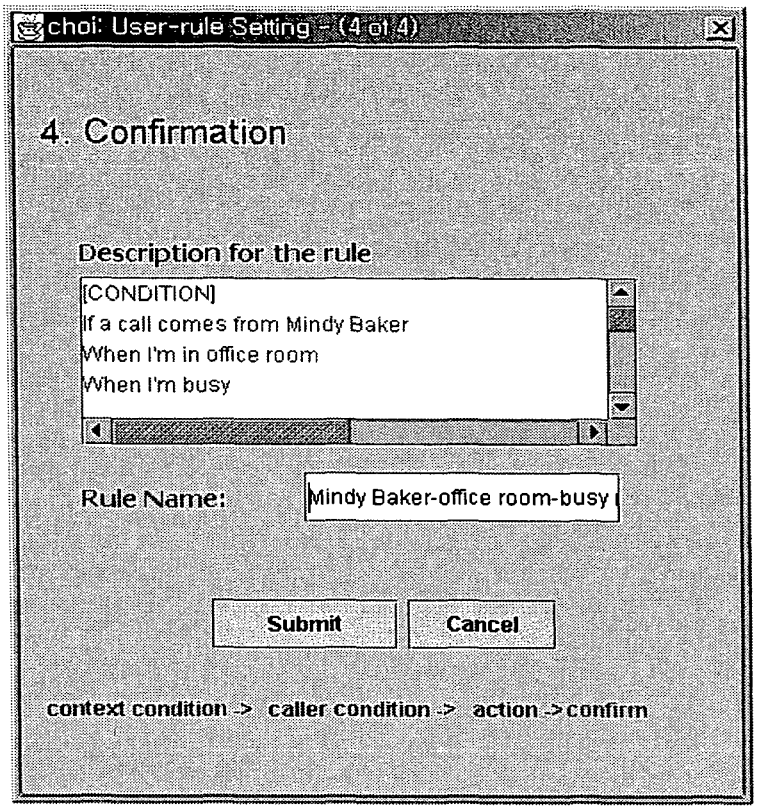


Figure 26

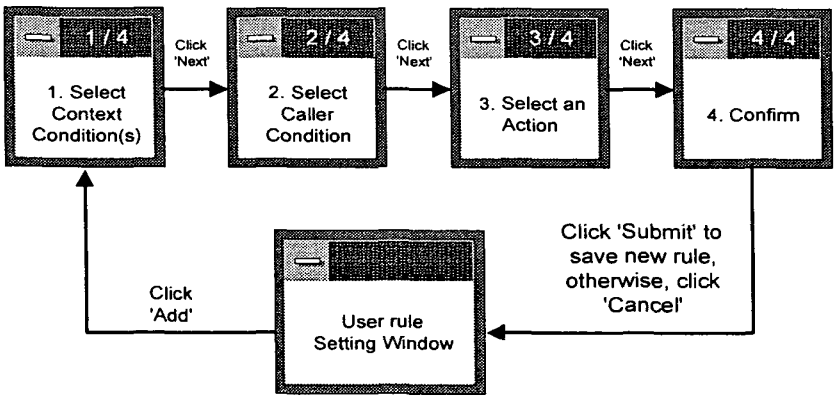


Figure 27

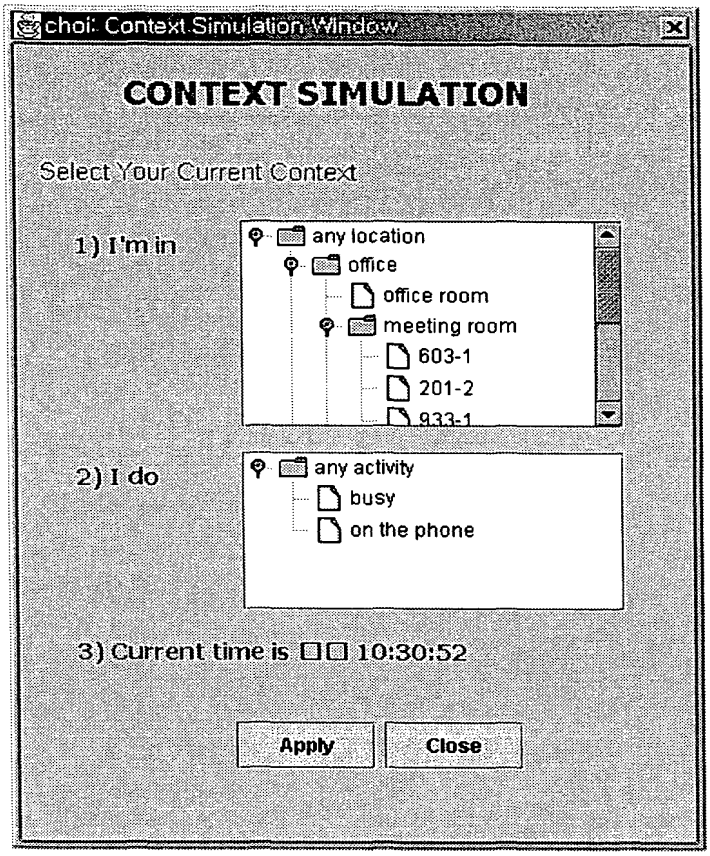


Figure 28

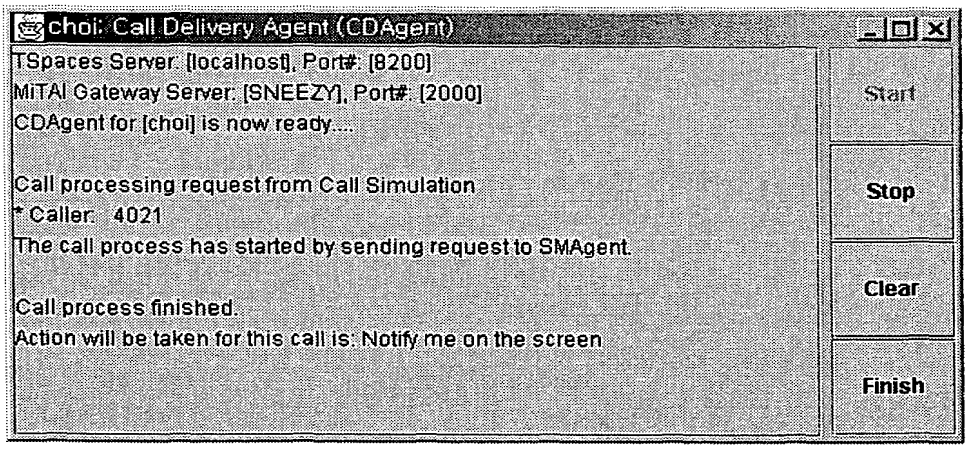


Figure 29

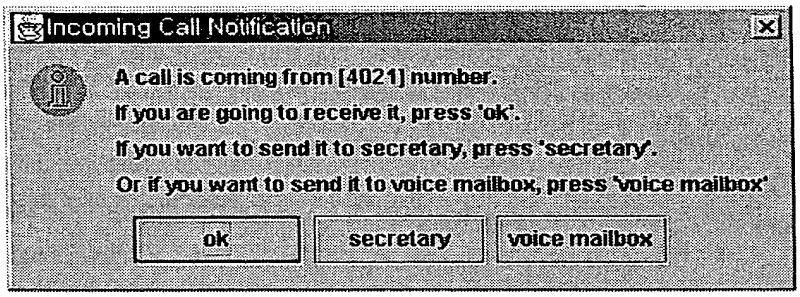


Figure 30

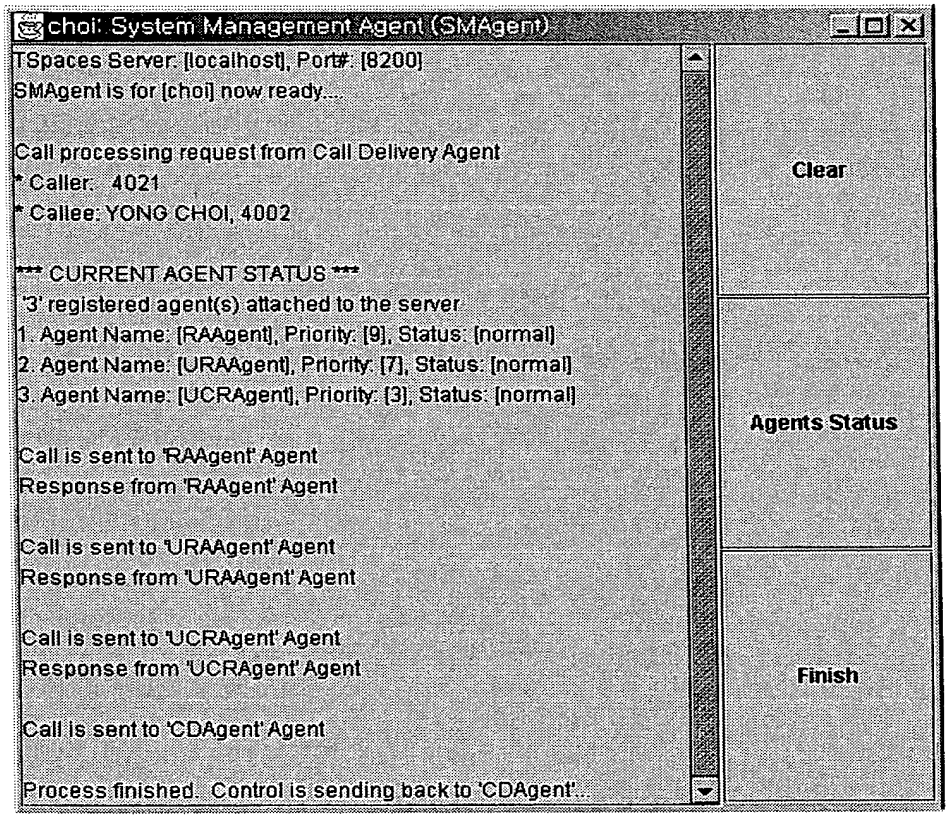


Figure 31